DATASETS, FEATURES, LEARNING, AND MODELS IN VISUAL RECOGNITION

BY

GANG WANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Professor David A. Forsyth, Chair
Assistant Professor Derek W. Hoiem
Professor Thomas S. Huang
Professor Narendra Ahuja
Associate Professor Mark A. Hasegawa-Johnson

ABSTRACT

Visual recognition is a fundamental research topic in computer vision. This dissertation explores datasets, features, learning, and models used for visual recognition.

In order to train visual models and evaluate different recognition algorithms, this dissertation develops an approach to collect object image datasets on web pages using an analysis of text around the image and of image appearance. This method exploits established online knowledge resources (Wikipedia pages for text; Flickr and Caltech data sets for images). The resources provide rich text and object appearance information. This dissertation describes results on two datasets. The first is Berg's collection of 10 animal categories; on this dataset, we significantly outperform previous approaches. On an additional set of 5 categories, experimental results show the effectiveness of the method.

Images are represented as features for visual recognition. This dissertation introduces a text-based image feature and demonstrates that it consistently improves performance on hard object classification problems. The feature is built using an auxiliary dataset of images annotated with tags, downloaded from the Internet. Image tags are noisy. The method obtains the text features of an unannotated image from the tags of its k-nearest neighbors in this auxiliary collection. A visual classifier presented with an object viewed under novel circumstances (say, a new viewing direction) must rely on its visual examples. This text feature may not change, because the auxiliary dataset likely contains a similar picture. While the tags associated with images are noisy, they are more stable when appearance changes. The performance of this feature is tested us-

ing PASCAL VOC 2006 and 2007 datasets. This feature performs well; it consistently improves the performance of visual object classifiers, and is particularly effective when the training dataset is small.

With more and more collected training data, computational cost becomes a bottleneck, especially when training sophisticated classifiers such as kernelized SVM. This dissertation proposes a fast training algorithm called Stochastic Intersection Kernel Machine (SIKMA). This proposed training method will be useful for many vision problems, as it can produce a kernel classifier that is more accurate than a linear classifier, and can be trained on tens of thousands of examples in two minutes. It processes training examples one by one in a sequence, so memory cost is no longer the bottleneck to process large scale datasets. This dissertation applies this approach to train classifiers of Flickr groups with many group training examples. The resulting Flickr group prediction scores can be used to measure image similarity between two images. Experimental results on the Corel dataset and a PASCAL VOC dataset show the learned Flickr features perform better on image matching, retrieval, and classification than conventional visual features.

Visual models are usually trained to best separate positive and negative training examples. However, when recognizing a large number of object categories, there may not be enough training examples for most objects, due to the intrinsic long-tailed distribution of objects in the real world. This dissertation proposes an approach to use comparative object similarity. The key insight is that, given a set of object categories which are similar and a set of categories which are dissimilar, a good object model should respond more strongly to examples from similar categories than to examples from dissimilar categories. This dissertation develops a regularized kernel machine algorithm to use this category dependent similarity regularization. Experiments on hundreds of categories show that our method can make significant improvement for categories with few or even no positive examples.

ACKNOWLEDGMENTS

I would like to give special acknowledgement to my thesis adviser, Professor David Forsyth. He consistently gives me priceless supervision and support. He is a perfect adviser with whom every graduate student wants to work.

I would like to thank Professor Derek Hoiem, with whom I worked closely. He not only gave me much guidance on developing big ideas, but also offered much help on improving details of the research.

I thank the other committee members for their insightful suggestions in my preliminary exam.

Thanks also go to my first graduate adviser, Professor Fei-Fei Li. She introduced me to computer vision and taught me many basic things.

Alexander Sorokin helped me a lot when I transferred to David's group. I learned much from numerous discussions with him.

Help and friendship from other vision group members and friends are also appreciated.

Last but most importantly, I want to thank all my family members for their vital support during my long period of study.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Visual recognition is a central topic of computer vision and includes both instance recognition (e.g., recognizing a dog as "Mike's dog" rather than other dogs) and category recognition (classifying a dog instance as dog rather than something else). The challenges of instance recognition include the change of viewpoint, cluttered background, and occlusions. A good survey on earlier approaches of instance recognition is [1]. Modern methods [2, 3] usually extract local features from images, match local features, and do geometric alignment to measure the similarity between two instances.

This dissertation focuses on category recognition, which is generally considered as more difficult because of intra-class variation. For example, different dogs may have different sizes, colors, and even shapes; different people may wear different clothes and have different poses such as sitting, standing, and riding bicycles. Many papers are published in top conferences and journals on category recognition. A recent survey paper on generic object recognition is [4]. Fei-Fei et al. [5] taught a short course on object recognition in CVPR and ICCV. This introduction surveys some category recognition papers in terms of datasets, features, and visual models.

Datasets

Datasets play an important role in research on category recognition. As pointed out by Ponce et al. [6], "They (datasets) are required for learning visual object models and for testing the performance of classification, detection, and localization algorithms." Collecting appropriate datasets is an important topic. We do not know what an ideal

dataset should be. In [7], we list several properties of a good dataset. For example, it should be rich enough to represent all major visual phenomena we are likely to encounter when applying the system to other images in the world; its labels should be reasonably accurate; it should cover many categories.

In the early stages, computer vision researchers created research datasets by taking pictures. Representative datasets are the ETH dataset [8], the COLT dataset [9], and the Caltech 5 dataset [10]. These images are taken in constrained environments and the number of subjects is limited, hence they cannot represent the variation of images in the real visual world.

One natural approach to create datasets is to download images from the Internet. But online images usually do not have clean ground truth labels that can be directly used. So researchers (usually graduate students) have to manually label images and delete noisy ones. Datasets from this category include the Caltech 101 [11], Caltech 256 [12], INRIA pedestrian [13], Scene 15 [14], and PASCAL VOC datasets [15]. Caltech 101 leads to a lot of competition [14, 16–21], since most previous datasets only cover a small number categories such as pedestrians and cars. Caltech 101 is also criticized for the lack of intra-class variability. Most Caltech 101 objects are big and in the center of images, and lack cluttered backgrounds. There is much more intra-class variation introduced in PASCAL VOC datasets [15], which enables them to be new benchmarks for evaluating recognition algorithms. Moreover, PASCAL images are annotated with object bounding boxes, segmentation masks, and person layout ground truth, so they can also be used to evaluate approaches on object localization, image segmentation, and person part detection.

But it is very time-consuming to select and label images from the Internet by a small number of labelers (e.g., graduate students), which prevents creating a large scale dataset. The most recent PASCAL VOC dataset only has around 10,000 images. Therefore, algorithms which can automatically find images relevant to an object are useful.

Li et al. [22] proposed an approach to incrementally train object models and collect object images with only visual appearance. Berg and Forsyth [23] showed that the associated text is very useful to suggest the relevance. Schroff et al. [24] used text features with strong semantics (image filename, image alt text and website title) rather than just nearby text. In [25], we exploit online Wikipedia resources to disambiguate queries; we also leverage object images from Caltech 101 and Caltech 256 to train object image models. By using online knowledge resources, we get significantly better performance than [23] and [24] on two test datasets. These automatic approaches usually work much better than commercial image search engines such as Google image search, but cannot produce reasonably accurate datasets. Human labeling is still needed to refine the annotation.

A strategy for building large scale datasets is to ask people on the Internet to label images. There are currently billions of Internet users among whom to recruit volunteers. LabelMe [26] is a public online image annotation tool, with which labelers can easily annotate segmentation polygons under instructions. But image labeling is very boring and tedious. The ESP game [27] and Peekaboom [28] are clever image annotation games: two random players annotate the same image with words, and get points if their annotations agree with each other. In this way, people are entertained and happy to annotate images; annotation quality is also guaranteed because of the agreement between two players. Some labeling tasks (e.g., labeling parts of humans), cannot be easily designed as games. Instead, Sorokin and Forsyth [29] employ people on Amazon Mechanical Turk to perform labeling tasks at low cost. Using the same idea, Deng et al. [30] create a large scale dataset called ImageNet, which provides whole image level category labels for more than three million images collected using several Internet search engines for about 5000 WordNet synsets.

Features

Visual recognition relies on sophisticated visual features. Feature detection was once

popularly adopted as a pre-processing step to extract features, which aims to find where and at what scale image features should be extracted. There are many criteria proposed. One is detecting local features in scale-space [31]. A feature point at a local maximum is selected in this 3D scale-space cube [32, 33]. Lowe [2] defines key locations as maxima and minima of the scale-space which is produced by convolving difference-of-Gaussian function with the image. The Kadir-Brady saliency detector [34, 35] is inspired by information theory principles. It aims to find salient regions, where the saliency is measured by patch pixel entropy as well as an additional criterion, the self-similarity. Matas et al. [36] find maximally stable extremal regions which are invariant in continuous transformation of image coordinates and monotonic transformation of image intensities. Maximally stable extremal regions are also found to result in better performance in stereo matching. Though feature detection is helpful in key-point detection and matching [37], Fei-Fei and Perona [38] show that densely sampled patches work much better than carefully selected regions in recognition. This is because densely sampled patches carry more visual information than sparse, detected regions. Following [38], recent image categorization and object recognition papers tend to adopt dense patches [14, 39].

We represent image patches with feature descriptors. Ideal recognition feature descriptors should be inter-class discriminative and intra-class repeatable. Gradient based features are proven to be most powerful in numerous studies. This family of features includes SIFT [2], HOG [13], PCA-SIFT [40], GLOH [41], and SURF [42]. Gradient based features work well mainly because they robustly capture edge information, which is important for recognition.

To create SIFT features, we first compute a gradient at each pixel, and quantize the orientation into bins (8 bins are found to work best in the empirical study of [2]). We then divide an image patch into cells (usually 4×4), and within each cell, we accumulate all pixels with their gradient magnitude according to their gradient orientation to

form a histogram. Histograms of all cells are concatenated to form the final descriptor. One pixel might also be close to other cells; in the spatial pooling procedure, bilinear interpolation is used to avoid boundary artifacts. GLOH uses a polar structure to quantize the spatial space, and has higher dimensions.

HOG features are an extension of SIFT features. The difference is that gradient strength is locally normalized by using bigger blocks from nearby regions. In contrast to fixed 4×4 cells in SIFT, HOG allows more flexible cell structures to best describe different categories in object detection.

PCA-SIFT is inspired by the use of gradient. It computes the x and y (gradient) derivatives and then reduces the resulting high-dimensional vector to 36 using principal component analysis (PCA). Similarly, SURF also operates on the x and y (gradient) derivatives; unlike PCA-SIFT, it simply summarizes them within different cells.

Another line of feature descriptor is shape features. Shape context [43] represents shape by counting the edge points within bins, which are normally taken to be uniform in log-polar space. This approach is in essence similar to SIFT, since edge points are usually extracted using gradient information.

Mikolajczyk and Schmid [41] compare feature performance on keypoint matching. In addition to SIFT, GLOH, and PCA-SIFT, they also test spin images [44], steerable filters [45], differential invariants [46], complex filters [47], and moment invariants [48]. SIFT is found to work best among these descriptors. A more recent and relevant comparison is conducted by Xiao et al. [49], where different feature descriptors are used to represent scene images for a classification task. HOG features leads to the best performance on hundreds of scene categories.

Visual Models

With extracted feature descriptors, we train visual models from training data, which are applied to predict class labels of test instances. The representation issue is how we represent objects/scenes based on a set of local features.

One approach simply ignores spatial relationships between local features (thus it is called "bag of features" similar to the term "bag of words" popularly used in natural language processing and information retrieval communities) which should have strong spatial correlation; e.g., people's heads are usually above their bodies in images. This approach achieves good performance since it can robustly and simply exploit all the extracted features, while approaches modeling spatial dependency usually have to get rid of features to be computationally tractable. The pioneering work on "bag of features" is conducted by Csurka et al. [50]. In their system, local image patches are extracted and represented as SIFT features. A subset of local features are quantized using k-means to form a visual vocabulary. Then each feature in an image can be represented as a word in the vocabulary. The global image representation is a histogram of visual words by summarizing all the local features. They train two classifiers on training data: one is the SVM classifier, and the other is the naive Bayesian classifier. SVM produces better results than naive Byesian. Zhang et al. [51] evaluate "bag of features" approaches with different feature detectors, different feature descriptors, and different SVM kernel functions. Their studies also show that SIFT works better than other features; the chi-square kernel and the earth mover's distance (EMD) [52] kernel work better than the linear kernel and RBF kernel.

Another notable work is by Sivic and Zisserman [53], in which video frames are represented as "bag of words" so that large scale retrieval and indexing is possible. An extension of "bag of words" models is topic models, which are originally studied in machine learning [54–56]. Topic models find word clusters according to their co-occurrence in documents, which are developed further in the computer vision community to discover objects in weakly labeled images and categorize object and scene images [17, 38, 57].

Simply disregarding all spatial layout information sacrifices descriptive ability, so Lazebnik et al. [14] find a good tradeoff to enforce geometric correspondence with a

spatial pyramid. Inspired by the spatial pooling idea of SIFT and GIST features [58], the approach does not explicitly model spatial relationships between local features, instead, it partitions the image into increasingly fine sub-regions and forms a histogram of visual words inside each sub-region. An SVM with a histogram intersection kernel follows to train the classifier. This approach shows much better performance than "bag of features" on scene categorization tasks.

There is also much study on modeling objects with parts and their geometric configuration. Biederman [59] shows that humans interpret images by components. One of the earliest computer vision works is pictorial structures [60], which is further developed by Felzenszwalb and Huttenlocher [61] for object recognition. It represents an object as a number of parts, whose appearance is modeled independently. The geometric configuration is represented as spring-like connections between pairs of parts. In recognition, it finds the best match of a pictorial structure model to an image with minimum energy. The authors also propose an efficient matching algorithm by restricting the spatial connections to a tree.

Similar to pictorial structures, constellation models [62–65] also model part appearance and spatial configuration between parts. But unlike pictorial structures, which only consider pairwise configuration, parts in constellation models are fully connected. The number of possible configurations increases exponentially with the number of parts. And it builds parts over single features, which makes it impractical to exploit many features. Much useful visual information is lost. Fergus et al. [66] relax the full connection and build a tree model. One part is selected as the landmark point, and the other parts are modeled independently given the landmark. The computation cost of this sparse object model is much reduced. Felzenszwalb et al. [67] extend this tree model for object detection. Instead of training a generative model as in [62, 66], a latent SVM classifier is trained in [67]. Positions and scales of object parts are considered as latent variables in training. They adopt a coordinate descent algorithm to infer

latent variables and update model parameters simultaneously. Their algorithm achieves the state-of-the-art performance on challenging object detection tasks in the PASCAL VOC challenge.

Crandall et al. [68] propose a K-fan model, where K denotes the number of landmark points and controls the tradeoff of representational power and the computational cost of doing inference with them. When $K = 0$, it is equivalent to a "bag of features" model, and no dependence between different object parts is considered; when K is the number of parts, it is equivalent to a fully connected constellation model. A hierarchical part based model is developed in [69]. Objects are modeled as parts and subparts. These models are developed under the concept of visual grammars [70, 71]. But they have not shown superior performance in real world datasets for recognition or detection.

Most existing approaches extract local features on grid image patches and train visual models over them. Gu et al. [72] attack recognition using segmented regions. They argue that there are two advantages of using region features: "(1) they encode shape and scale information of objects naturally; (2) they are only mildly affected by background clutter." But regions are sensitive to segmentation errors. They segment images using a recently developed robust segmentation algorithm. Each region is represented by a rich set of image cues (shape, color and texture). Region weights are learned using a max-margin framework for recognition. Promising results are shown on benchmark datasets such as Caltech 101. Image regions are more likely to include a coherent object than rigid image patches. Russell et al. [73] propose to discover objects from a set of image segments.

Above, I introduced a number of approaches which aim to predict class labels of images or localize object instances using bounding boxes. Shotton et al. [74] try to classify each pixel in images. They train discriminative object class models with texture-layout filters, which model texture patterns and their spatial layout together. Segmentation

is performed using a conditional random field model. Shotton et al. [75] propose an efficient system using semantic texton forests, which are efficient features by applying forests of decision trees on image pixels; hence the expensive computation of filter-bank responses in [74] is avoided.

This dissertation describes my work on visual recognition during my PhD study. It presents four pieces of work which attack four important issues in recognition: datasets, features, learning, and models. Current datasets are usually collected by manually labeling, which is expensive even using Amazon Mechanical Turk. We develop an approach to automatically collect object images from the Internet to form object datasets. It exploits online knowledge resources as training data to train object image and text models. Wikipedia is used as a text resource; Caltech 101 and 256 datasets are used as an image resource. Our approach shows superior performance on a benchmark dataset and can be applied to collect large scale datasets (manual identification might be needed). We also propose a new type of feature called text features. We build text features by matching an input image with Internet images and summarizing the text of its nearest neighbors. Text features more directly reflect the semantics of the scene and are complementary to visual features. Better recognition performance can be achieved by combing the two features. With larger and larger datasets, the computational cost becomes a new challenge. We develop a fast learning algorithm to learn SVM classifiers with a histogram intersection kernel. It processes training examples one by one in a sequence and is memory efficient. Our algorithm makes it possible to train classifiers with many training examples. Current statistical models cannot handle categories with few or no positive training examples. We develop a model to transfer visual knowledge from labeled similar and dissimilar categories. Its effectiveness is shown on hundreds of novel object categories. The following chapters introduce these contributions in detail.

CHAPTER 2

COLLECTING OBJECT IMAGES USING ONLINE KNOWLEDGE RESOURCES

To train visual models and evaluate recognition algorithms, we need to collect object images as training data. We collect images from the Internet for object categories by using both text and image cues. For a given object category, we train its text and image models with online knowledge resources (Wikipedia for text; Caltech 101 and Caltech 256 for images). Our approach is automatic, meaning one could build a search engine that took text queries, extracted information from the knowledge resources, and identified relevant images — except for a step where we identify the sense of the query word (which we do by offering a user a set of senses from Wikipedia). The framework of our approach is shown in Figure 2.1. We perform experiments on the same animal dataset of [23], which includes 10 animal categories. Our method outperforms reported results [23, 24]. We collect five new categories: "binoculars," "fern," "laptop," "motorbike" and "rifle." The experimental result shows our algorithm can also get high precision over these categories.

Our work is related to:

Words and Pictures: There are many datasets of images with associated words. Examples include: collections of museum material [76]; the Corel collection of images ([77, 78], and numerous others); any video with sound or closed captioning [79]; images collected from the web with their enclosing web pages [23]; or captioned news images [80]. It is a remarkable fact that, in these collections, pictures and their associated annotations are complementary. The literature is very extensive, and we can

mention only the most relevant papers here. For a more complete review, we refer readers to [81]. Joint image-keyword searches are successful [82], and one can identify images that illustrate a story by search [83]. Clustering images and words jointly can produce useful, browsable representations of museum collections [76].

Linking keywords to images: One could predict words associated with an image (*image annotation*) or predict words associated with particular image structures (*image labeling*). Because words are correlated, it can be helpful to cluster them and predict clusters, particularly for annotation [84]. Labeling methods are distinguished by the way correspondence between image structures and labels is inferred. Methods include: clustering or latent variable methods [76, 77], using multiple-instance learning [78, 85], explicit correspondence reasoning with generative models [86,87], latent Dirichlet allocation [88], cross-media relevance models [89], continuous relevance models [90], and localization reasoning [91]. Barnard et al. [77] demonstrate and compare a wide variety of methods to predict keywords, including several strategies for reasoning about correspondence directly. Most methods attempt to predict noun annotations, and are more successful with mass nouns — known in vision circles as "stuff"; examples include *sky*, *cloud*, *grass*, *sea* — than with count nouns ("things": *cat*, *dog*, *car*). For these methods, evaluation is by comparing predicted annotations with known annotations. Most methods can beat a word prior, but display marked eccentricities. One could then propagate text labels from labeled images to unlabeled images, making keyword based searches of large image collections possible.

2.1 Approach

As stated before, our goal is to retrieve object images from noisy web pages with image and text cues. We have a query q which is the object class name, for example, "frog." We also have a collection of web pages which are collected by inputting q and some extensions to the Google text search engine. The ith web page is represented as a

packet $\{W_i, I_i\}, i = 1, \cdots, N$, where I_i denotes image and W_i denotes text nearby I_i. We write $c_i = 1$ if I_i is relevant to q; otherwise $c_i = 0$. We write θ_t for the text model parameter and θ_v for the image model parameter when $c_i = 1$; write θ_b for the text model parameter when $c_i = 0$. We rank images according to

$$p(c_i = 1 \mid W_i, I_i, q; \theta_t, \theta_v, \theta_b) \tag{2.1}$$

We adopt a generative text model and a discriminative image model. Equation 2.1 is written as

$$\frac{p(W_i \mid c_i = 1, q; \theta_t) p(c_i = 1 \mid I_i, q; \theta_v)}{p(W_i \mid I_i, q)} \tag{2.2}$$

where $p(W_i \mid I_i, q)$ is

$$\begin{aligned} &p(W_i \mid c_i = 1, q; \theta_t) p(c_i = 1 \mid I_i, q; \theta_v) + \\ &p(W_i \mid c_i = 0, q; \theta_b) p(c_i = 0 \mid I_i, q) \end{aligned} \tag{2.3}$$

where $p(c_i = 0 \mid I_i, q)$ equals $1 - p(c_i = 1 \mid I_i, q)$.

Parameters θ_t and θ_v are trained on text and image knowledge resources. Figure 2.1 takes the query "frog" as an example to illustrate our approach. We show how to learn $p(W_i \mid c_i = 1, q; \theta_t)$ and $p(W_i \mid c_i = 0, q; \theta_b)$ in Section 2.1.1, and $p(c_i = 1 \mid I_i, q; \theta_v)$ is studied in Section 2.1.2.

2.1.1 Text model

We adopt a generative text model. W_i is a sequence of words $\{w_i^j, j = 1, \cdots, L\}$. θ_t is multinomial parameter over words and is estimated from the text knowledge resource. Assume words are independent from each other in W_i:

$$p(W_i \mid c_i = 1, q; \theta_t) = \prod_{j=1}^{L} p(w_i^j \mid c_i = 1, q; \theta_t) \tag{2.4}$$

But Equation 2.4 tends to underweight the contribution of long text. For example, a short sentence may be accidental, but a paragraph is not. So we use the following

formula:

$$p(W_i \mid c_i = 1, q; \theta_t) = (\prod_{j=1}^{L} p(w_i^j \mid c_i = 1, q; \theta_t))^{\frac{1}{L}} \tag{2.5}$$

which weights longer sets of relevant text more heavily in posterior inference (see Equation 2.2).

The text knowledge resource is denoted K. It is a simple combination of all the Wikipedia pages (just body text) from the queried object class (with desired sense) and its descendant classes in the Wikipedia taxonomy. In the simplest case, θ_t could be estimated from K by maximum likelihood, which estimates θ_t^j (the jth component of the multinomial parameter) as a ratio of word counts. However, the word set of K is limited, meaning that zero counts are a problem, so "smoothing" is necessary. In this chapter, we adopt Dirichlet smoothing [92] since it is simple and effective.

A much richer Wikipedia page collection A is extracted. The pages are from a number of semantically close classes (except children classes) of the object. With A as smoothing data, θ_t is estimated as

$$\theta_t^j = \frac{N_K^j + \lambda \eta^j}{N_K + \lambda} \tag{2.6}$$

where N_K^j denotes the counts of the jth word in K and N_K denotes the counts of all the words in K. Similarly, $\eta^j = \frac{N_A^j}{N_A}$, and λ is a parameter to control the contribution of the prior. Words are set to be independent and of uniform probability when $c = 0$. Then $p(W_i \mid c_i = 0, q; \theta_b)$ is calculated similarly to Equation 2.5.

2.1.2 Image model

We use a discriminative method to learn $p(c_i = 1 \mid I_i, q; \theta_v)$ directly. An SVM is employed because it has been proven to be effective and highly robust to noise in image classification [14,24]. We exploit Caltech or Flickr images of the queried object class as positive training examples; images from the "clutter" category of Caltech 256 are used as negative examples. Each image is represented as a normalized histogram of visual

words with dimension l. Training examples for this task are denoted as $\{(x_r, y_r), r = 1, \cdots, R, y_r = 1, -1\}$.

The original SVM classifier just outputs a hard decision. In this paper, we adopt the method of [93] to fit a posterior probability with a sigmoid function.

Dimension l is usually high, and it is difficult to learn the model in the high-dimensional space. References [94, 95] have shown that using subsidiary tasks can produce a low-dimensional feature space, which is stable and effective for the problem at hand. But instead of using unlabeled data as the subsidiary task, we propose a novel method to exploit highly relevant images in the knowledge resources, which are more helpful for the main task.

We first represent the object class we want to query as a normalized histogram of codewords f_o by using all positive training images. Other categories from the Caltech dataset (except the queried object class) are also represented as histograms $f_m, m = 1, \cdots, M$. We calculate the difference between queried object class and Caltech classes using χ^2:

$$\frac{1}{2}\sum_{j=1}^{l}\frac{[f_o^j - f_m^j]^2}{f_o^j + f_m^j} \tag{2.7}$$

The T_s most similar categories are chosen from Caltech and act as positive examples in the subsidiary tasks. We download T_n sets of background images from the web as negative examples. By pairwise matching, there are $T = T_s T_n$ subsidiary tasks overall. Each image in subsidiary tasks is also represented as a normalized histogram with dimension l. Similar to [95], for each auxiliary task t, we learn a linear function w_t^* which is most discriminative between positive and negative training images with a linear SVM.

We concatenate all w_t^* (each w_t^* is a column) to form a matrix W with dimension $l \times T$. We obtain a projection matrix P with dimension $h \times l$ by taking the first h eigenvectors ($h \ll l$) of matrix WW'. The training examples for the main task are

now represented in the new feature space as $\{(P \cdot x_r, y_r), r = 1, \cdots, R, y_r = 1, -1\}$, where $P \cdot x_r$ is with dimension h. A kernel SVM classifier with optimal parameter w^* is trained:

$$\min \quad \|w\|^2 + C \sum_r \xi_r \tag{2.8}$$

$$\text{subject to} \quad y_r(w \cdot \Phi(P \cdot x_r) - b) \geq 1 \tag{2.9}$$

where Φ denotes the kernel function. We use a radial basis function in this chapter. To calculate $p(c_i = 1 \mid I_i)$, I_i is represented in the low-dimensional feature space and the learned kernel SVM classifier is applied. SVM decision is converted to a probability with the method of [93].

The overview of our learning algorithm is presented in Alg.1.

Algorithm 1 Overview of image model learning.

For a given query:

1. **Obtaining training examples**: Use Caltech or Flick images of the queried object class as positive training examples; use "clutter" category from Caltech 256 as negative training examples.

2. **Representation**: Represent image as a normalized histogram of codewords with dimension l; represent queried object class and other categories in Caltech datasets as normalized histograms with dimension l too.

3. **Constructing subsidiary tasks**: By chi-square measure over histograms, find the T_s most similar categories from Caltech dataset and set them to be positive examples in subsidiary tasks. Download T_n sets of background images from the web as negative examples. By pairwise matching, there are $T = T_s T_n$ subsidiary tasks overall.

4. **Learning feature projection**: For each subsidiary task t, learn linear function w_t^* with linear SVM. Concatenate all w_t^* to form a matrix W with dimension $l \times T$. By taking the first h eigenvectors of WW', get a projection matrix P with dimension $h \times l$.

5. **Training SVM classifier**: Convert training examples of the main task to low-dimensional space with projection matrix P; train a kernel SVM.

2.2 Implementation

In this section, we give implementation details for the text and image models. In Equation 2.6, λ is set to be one tenth of the words number of K. We remove stop words from Wikipedia and collected web pages.

Both training and testing images are converted to gray scale and resized to a moderate size. We use a canny edge detector to extract edge points from images. A set of points are randomly selected and regions are extracted at these points. As in [96], scale is uniformly sampled from a sensible range (10-30 pixels in this paper). Around 400 regions from each image are extracted. We represent these regions with SIFT [97] features. Features from 150 Caltech categories (100 categories from Caltech101 plus 50 from Caltech 256) are quantized with K-means. The number of clusters is 500, so each image and class is represented with a 500-dimensional histogram.

When constructing subsidiary tasks, the 10 most similar categories are selected out, and 3 sets of background images are downloaded from web. So there are 30 subsidiary tasks by pairwise matching. We reduce the 500-dimensional feature to be 20-dimensional.

As pointed out by [24], there are "abstract" images which do not look realistic, such as comics, graphs, plots and charts. In order to get natural images, it is better to remove them. In their work, they learned an SVM classifier between "abstract" and "non-abstract" with extra training images. In this dissertation, we simply remove the non-color images since most of the "abstract" images are black and white.

2.3 Experiments

We perform two experiments in this dissertation. The first one is on the dataset of [23], which includes 10 animal classes as shown in Figure 2.2. The second experiment is performed on five newly collected categories.

Besides the combined model in Equation 2.1, we also perform retrieval experiments with a pure text model and a pure image model. The text model ranks images according to

$$\frac{p(W_i \mid c_i = 1, q; \theta_t)p(c_i = 1, q)}{p(W_i \mid c_i = 0, q; \theta_b)p(c_i = 0, q)} \tag{2.10}$$

where $p(c_i = 1, q)$ and $p(c_i = 0, q)$ are simply set to be equal.

The image model ranks images with $p(c_i = 1 \mid I_i, q; \theta_v)$.

2.3.1 Experiment 1

For each animal class, we use its Wikipedia pages and Caltech or Flickr images as knowledge resources to train the text and image models. Then images in the web pages returned by Google are ranked for each class. There is no "monkey" in the Caltech data sets, so we use Flickr images to train the "monkey" image model. For the other nine categories, we use Caltech images. We also compare the performance with different types of training images in Figure 2.2, which shows precision at 100 image recall by the pure image model. "Flickr" denotes that the image model is trained with noisy Flickr images as positive training examples. Similarly, "Caltech" denotes that the model is trained with Caltech images; and "Flickr and Caltech" denotes that the model is trained with both Flickr and Caltech images. In most categories, clean Caltech images produce better results. Results using Flickr images are comparable and acceptable, which shows we can use Flickr images to train the image model if there are no clean Caltech images available.

In Figure 2.3, we present precision recall curves with different models. In all figures, the x axis denotes recall while the y axis denotes precision. "Text" is the result with the text model; "Image" is the result with the image model. "Text+Image" shows the result with the combined model. Note that we do not remove "abstract" images here.

We compare our ranking results produced by the combined model with the work of [24] and of [23] in Figure 2.4. This is based on the precision of 100 image recall. Note that we use the result of "classification on test data" in [23]. We outperform [23] on all the categories and outperform [24] except on "Alligator" and "Beaver." Improvement is significant for categories such as "Bear," "Dolphin," "Monkey" and "Penguin." We also make a comparison with different measures on the whole dataset as shown in Table 2.1.

We show the top-ranked images for "Alligator," "Bear," "Frog," "Dolphin," "Giraffe," "Penguin" and "Leopard" in Figure 2.5. Images in the red squares are false positives. Most of these images are correct.

2.3.2 Experiment 2

Experiment 1 is carried out only on animal categories. In this section, we collect five diverse object classes ("binoculars," "fern," "laptop," "motorbike" and "rifle"). Similar to [23], we query Google with the object name and some extensions. The top returned web pages are collected. We restricted downloaded images to be JPEG format. Finally, we get 732 "binoculars" images, 323 of which are correct images; 501 "laptop" images, 158 of which are correct; 636 "fern" images, 190 of which are correct; 801 "motorbike" images, 276 of which are correct; and 921 "rifle" images, 195 of which are correct.

Similar to Experiment 1, our algorithm is applied to these categories and the precision recall curves are shown in Figure 2.6. In Table 2.2, we show the precision at 100 image recall with different models. Highly ranked images are exhibited in Figure 2.7. False positive images are marked with red squares.

2.4 Tables and Figures

Table 2.1: Overall comparison with Schroff et al. [24] and Berg and Forsyth [23] on the 10 animal categories. This is based on the precision at 100 image recall. Our method outperforms them on all four measures: Mean, Median, Minimum, and Maximum.

	Mean	Median	Minimum	Maximum
[23]	55.1	61	15	83
[24]	63.3	64	36	88
Our result	**79.4**	**84**	**41**	**94**

Table 2.2: Precision at 100 image recall. "Text" is the result with text model; "Image" is the result with image model. "Text+Image" shows the result with combined model.

	Text	Image	Text+Image
Binoculars	76	90	93
Laptop	58	41	67
Fern	72	68	80
Motorbike	57	34	63
Rifle	55	21	57

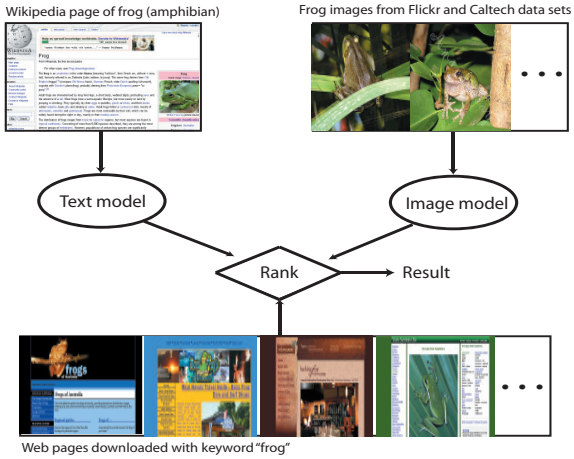

Figure 2.1: The framework of our approach. The query "frog" is taken as an example in this figure. We collect a pool of noisy web pages by inputting "frog" to Google. The Wikipedia page of frog (amphibian) is extracted and a text model is built with its textual description. Similarly, the image model is trained with Caltech and Flickr "frog" images. By combining text and image cues, images from web pages are ranked.

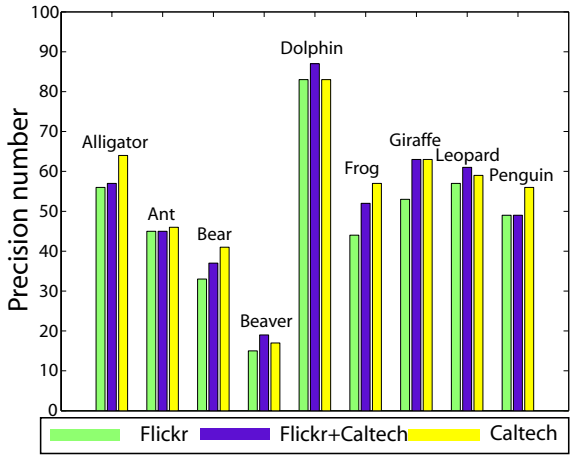

Figure 2.2: Precision at 100 image recall by image model. "Flickr" denotes that the model is trained with Flickr images as positive training examples. Similarly, "Caltech" denotes that the model is trained with Caltech images, and "Flickr and Caltech" denotes that the model is trained with both Flickr and Caltech images. In most categories, clean Caltech images produce better results. But results by Flickr are comparable and acceptable. This shows we can build the image model with Flickr if there are no clean Caltech images for the queried object class.

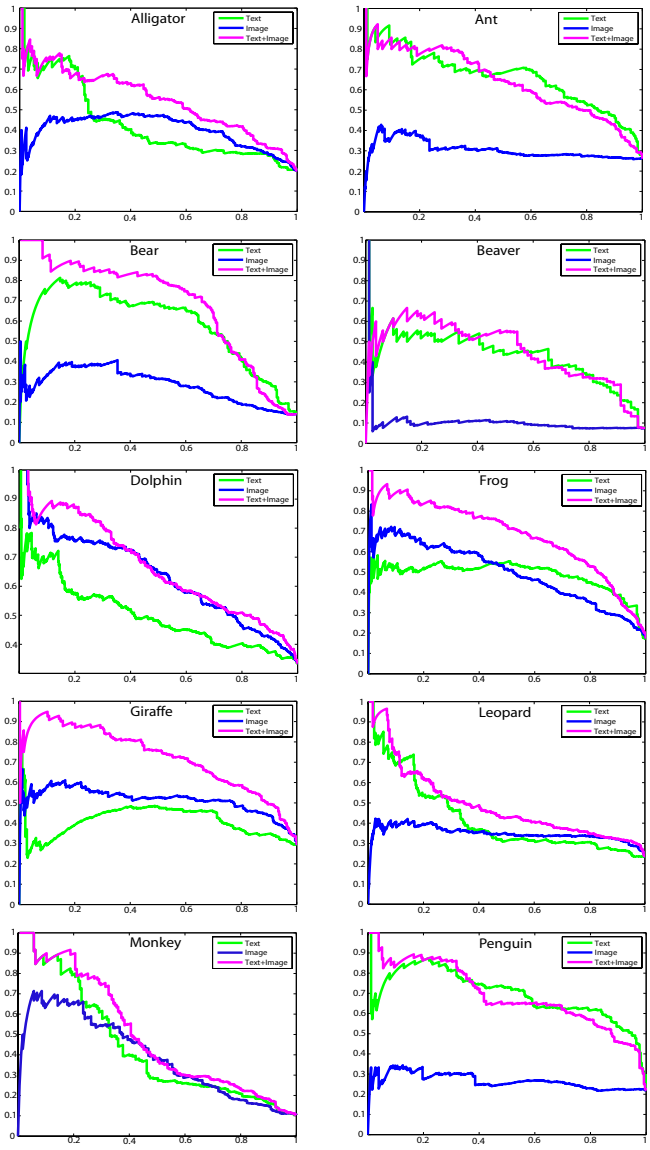

Figure 2.3: Precision recall curves with different models. In all figures, x axis denotes recall while y axis denotes precision. "Text" shows the result with text model; "Image" shows the result with image model. "Text+Image" shows the result of combining text and image models. Note that we do not remove "abstract" images here. The combined model usually works better than separate models. Image models can be quite discriminative, such as the "dolphin" image model and the "frog" image model.

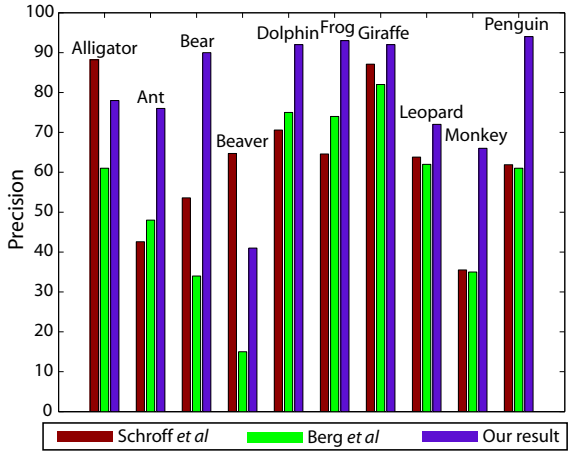

Figure 2.4: Results comparison with previous papers for each category. This is based on the precision on 100 image recall. We outperform them over all the categories except "alligator" and "beaver." Improvement over many categories is significant, such as "bear," "dolphin," "monkey," and "penguin."

Figure 2.5: Top-ranked images for “alligator,” “bear,” “frog,” “dolphin,” “giraffe,” “penguin,” and “leopard.” Images in red squares are false positives. Most of the images are correct.

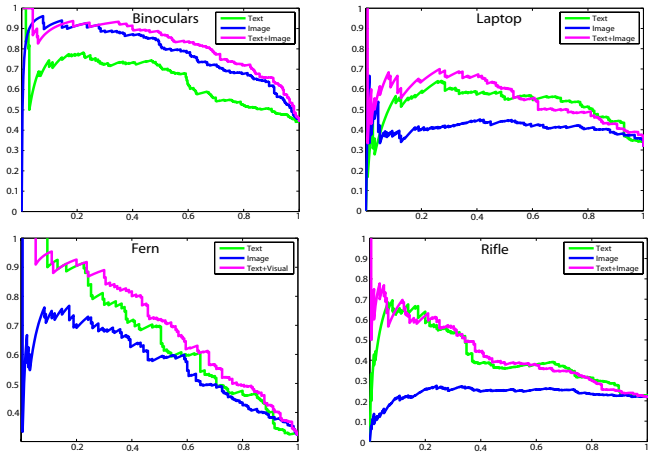

Figure 2.6: Precision recall curves with different models. In all figures, x axis denotes recall while y axis denotes precision. "Text" shows the result with text model; "Image" shows the result with image model. "Text+Image" shows the result with the combined model.

Figure 2.7: Top-ranked images for "binoculars," "laptop," "fern," and "rifle." Images in red squares are false positives.

CHAPTER 3

BUILDING TEXT FEATURES FROM THE INTERNET FOR VISUAL RECOGNITION

In this chapter, we introduce an approach to build text features from the Internet for object image representation. The idea to build such text features is illustrated in Figure 3.1.

3.1 Approach

Our approach is to build text features for object image classification. The text features are expected to capture the semantic meaning of images and provide a more direct gateway to image analysis. Figure 3.2 shows the feature extraction and classification procedure. We have a dataset with training and test images. We also have an auxiliary dataset of Internet images (downloaded from Flickr), which have associated text. For each training image, we extract visual features and find its K nearest neighbor images from the Internet dataset. Text associated with these near neighbor Internet images is used to build the text features. Text classifiers are then trained on the text features. For a test image, we follow the same procedure to construct its text features, and use the trained text classifiers to predict the category labels. We also train a separate classifier on the visual features. We obtain the final prediction from a third classifier trained on the confidence values returned by the text and the visual classifiers.

3.1.1 Visual features

We use five types of features to find the nearest neighbor images and train visual classifiers.

The SIFT feature [2] is popularly used for image matching and object recognition. We use it to detect and describe local patches. We extract about 1000 patches from each image. The SIFT features are quantized to 1000 clusters and each patch is denoted as a cluster index. Each image is then represented as a normalized histogram of the cluster indices.

The Gist feature has been proven to be very powerful in scene categorization and retrieval [58]. We represent each image as a 960-dimensional Gist descriptor.

We extract Color features in the RGB space. We quantize each channel to 8 bins, then each pixel is represented as an integer value range from 1 to 512. Each image is represented as a 512-dimensional histogram by counting all the pixels. The histogram is normalized.

We also extract a very simple Gradient feature, which can be considered as a global and coarse SIFT feature. We divide the image into 4*4 cells, and at each cell we quantize the gradient into 16 bins. The whole image is represented as a 256-dimensional vector.

The Unified feature is a concatenation of the above four features. We learn weights for different feature types to make the unified feature discriminative. Writing the four features introduced above as f_1, f_2, f_3 and f_4 respectively, our new feature is the concatenation of $w_1 f_1$, $w_2 f_2$, $w_3 f_3$ and $w_4 f_4$, where w_j is a non-negative number to indicate the importance of the jth feature.

We learn the weights from the training images. We aim to force the images from the same category to be close, and images from different categories to be far away in the new feature space. We randomly select N pairs of images from the training set. For the ith pair, $S_i = 1$ if the two images share at least one same object class; otherwise, $S_i = 0$. We calculate the chi-square distance with f_j for the ith pair as d_i^j. Then we

learn feature weights by minimizing the following objective function:

$$\sum_i (e^{-\sum_j w_j d_i^j} - S_i)^2 \tag{3.1}$$

This optimization problem can be straightforwardly solved using the "fmincon" function in MATLAB.

3.1.2 Internet dataset

The auxiliary Internet dataset provides association between images and text. With this dataset, we can build text features for the images which do not have text by nearest neighbor matching.

The Internet is rich in multimedia, and there is strong correlation between images and text. This is especially apparent in the photo sharing web sites such as Flickr: users tag images with some keywords, which usually describe the visual content of the images. Users also group images by the content. For example, there is a group called "Dogs! Dogs! Dogs!" which contains dog images. The group name becomes a very strong text cue to indicate the visual content of the images.

Our auxiliary dataset is collected from Flickr, and consists of about 1 million images. About 700,000 images are collected for 58 object categories, whose names come from PASCAL categories such as "car" or Caltech 256 [12] such as "penguin" and "rainbow." The other images are collected from a group called "10 million photos." These images are drawn from random categories.

3.1.3 Text features

Once the text features are extracted from the auxiliary dataset, they represent the image in a way that more directly reflects the semantics.

For each training and test image in our dataset, we find its K nearest neighbor images from the auxiliary dataset with the visual features. The text associated with these nearest neighbor images is extracted to build the text features. We treat each tag

and group name as an individual item in our text feature representation, even though it may include multiple words. For example, the group name "Dogs! Dogs! Dogs!" is treated as a single item. We only use a set of frequent tags and group names (about 6000) in the auxiliary dataset. The other tags and group names are not counted. The text feature is a normalized histogram of tag and group name counts.

3.1.4 Classifier

The purpose of this chapter is to show that a text feature, computed from the auxiliary dataset, is in fact a powerful and general descriptor. Various classifiers could be applied to such a feature. We have chosen to use an SVM classifier with a chi-square kernel for the text features. The same classifier is used for the visual features.

3.1.5 Fusion

We now have two types of features: the standard visual features and the text features. We do not believe there is likely to be much interaction, in the sense that one feature can tell when the other is unreliable. Therefore, we build two separate classifiers, one for the text features and the other for the visual features. A third classifier is then trained to combine the confidence values of the two initial classifiers into a final prediction. This final classifier uses logistic regression and is trained on a validation set.

3.2 Experiment

We perform image classification experiments on two datasets: PASCAL VOC 2006 and PASCAL VOC 2007. The PASCAL 2006 dataset has 10 object categories while the 2007 dataset has 20 categories. The 2007 dataset is more difficult because there is much more variation with the object appearance. To ensure that there were no PASCAL test images in our auxiliary Internet dataset, we removed all images from the auxiliary set that had a small image distance (within a threshold) to any image in the test set.

According to the standard evaluation measure, the performance is quantitatively measured by AUC (area under the ROC curve) value on the 2006 dataset and measured by AP (average precision) value on the 2007 dataset. When evaluating the methods, we are interested in the following phenomena: (1) performance of text features which are built with different visual features, (2) effects of combining text and visual classifiers, (3) effects of varying number of training images, (4) performance of the text features built with varying numbers of Internet images, (5) effects of the category names.

3.2.1 Results: Text features built with different visual features

We could use different types of visual features to retrieve the nearest neighbor images to build the text features. We use 150 nearest neighbor images in all the experiments. The performance on 2006 and 2007 for each object category is listed in Table 3.1 and Table 3.2 respectively. We use a KNN classifier as a baseline in Table 3.1 for the 2006 dataset. Each Internet image is considered to be a positive example of the object categories whose names appear in the associated text. Then a test image can be simply classified by the KNN classifier. Our text classifier significantly outperforms KNN for each individual feature.

The performance of the text features is affected by the strength of the visual features. The better KNN performs, the better the text features are. This is because good visual features can find good nearest neighbor images to build good text features. So the text features built by the unified visual features usually work best and the text features built by the color features usually work worst on both of the datasets.

3.2.2 Results: Combining text and visual classifiers

Text features do not outperform visual features as shown in Table 3.1 and Table 3.2. But text features are quite different from visual features, so they can correct each other, and the combination should result in improvement.

Table 3.1 and Table 3.2 show that the combination consistently outperforms sep-

arate classifiers (the best performance in each panel is indicated in bold; look for the bold horizontal line). In Figure 3.3, we show several examples which are misclassified by the visual classifier, but correctly classified by the text classifier on PASCAL 2006. The objects vary widely. In the first image, the cat is in a playful pose, which is unusual in the PASCAL training set. So the visual classifier gets it wrong. However, we may find many such images in the auxiliary dataset (there are several sleeping cat images in the 25 nearest neighbors). Now the text cue can make a correct prediction. The text vector also shows that the group name is an important cue. There are several peaky groups such as "somebody else's cat," "all animals" and so on. In Figure 3.4, we also show images which are misclassified by the text classifier but correctly classified by the visual classifier. This happens when we fail to find good nearest neighbor images.

At the bottom of Table 3.1 and Table 3.2, we show the performance obtained by combining the different classifiers, which is achieved by training a logistic regression classifier on the validation dataset using the confidence values returned by the individual classifiers as features. Combining all the visual classifiers works better than combining only visual classifiers or text classifiers.

3.2.3 Results: Varying numbers of training images

In Figure 3.5, we show the performance with different numbers of training images on PASCAL 2006. We randomly select $1/40$, $1/30$, $1/20$, $1/10$, $1/5$, $1/2$ of the positive and negative images, respectively, in the training data for each category to do the experiments. For comparison, we also show the results with all the training images. The performance is shown by the average AUC values over all the categories. We do experiments by the "Gist" and the "Combination" of multiple classifiers. We observe that the text features outperform the visual features when there are only a small set of training images available. There is always improvement by combining the two types of features, but the gain is insignificant when the two classifiers are not comparable.

3.2.4 Result: Varying numbers of auxiliary images

We also test the performance of the text features built with varying numbers of Internet images in Table 3.3 on PASCAL 2006. We randomly select 200,000 and 600,000 images from the collection to build the text features. The result is based on the average AUC values over the 10 object categories.

Increasing the image number from 200,000 to 600,000 leads to a big improvement, but further increasing to 1 million results in a negligible improvement.

This means that merely increasing the size of the auxiliary dataset may not have much impact. Instead, one should create an auxiliary dataset covering more meaningful images and improve the technique to find good nearest neighbor images.

3.2.5 Result: Excluding the category names

Our text features might be powerful only because our images are tagged with category labels. To test this, we exclude category names and their plural inflections from the text features. This means that, for example, the words "cat" and "cats" would not appear in the features. The effect on performance is extremely small (Table 3.4). This suggests that text associated with images is rich in secondary cues (perhaps "mice" or "catnip" appear strongly with cats). In future work, we will investigate directly applying semantic measures of similarity to our features.

3.3 Tables and Figures

Table 3.1: The AUC values with different settings on PASCAL 2006 for each object category. Take the Gist feature as an example: "Gist(KNN)" denotes the result with a KNN classifier using the Gist feature; "Gist(V)" denotes the result with the visual SVM classifier; "Gist(T)" denotes the result with the text SVM classifier; "Gist(T+V)" denotes the result by fusing the outputs of the text and visual SVM classifiers. Our text classifier outperforms the KNN classifier. The performance of the text features depends on the strength of the visual features. "Unified(T)" usually works best among all the text classifiers and "Color(T)" usually works worst. We can get better performance in almost all of the categories by combining the text and visual classifiers. The results by combining all the text classifiers, all the visual classifiers and all the text and visual classifiers, are indicated by "Combination(T)," "Combination(V)" and "Combination(T+V)" respectively.

	bicycle	bus	car	cat	cow	dog	horse	motorbike	person	sheep
Gist(KNN)	0.795	0.875	0.885	0.736	0.820	0.674	0.734	0.822	0.605	0.868
Gist(V)	0.825	0.951	0.940	0.861	0.876	0.773	0.845	0.862	0.762	0.914
Gist(T)	0.818	0.915	0.932	0.812	0.843	0.744	0.820	0.878	0.733	0.875
Gist(V+T)	**0.837**	**0.955**	**0.941**	**0.869**	**0.880**	**0.790**	**0.858**	**0.886**	**0.769**	**0.917**
Gra(KNN)	0.734	0.837	0.902	0.743	0.808	0.666	0.743	0.786	0.625	0.799
Grad(V)	0.826	0.933	0.944	0.861	0.842	0.746	0.825	0.863	0.743	0.870
Grad(T)	0.810	0.931	0.935	0.806	0.830	0.725	0.776	0.817	0.722	0.855
Grad(V+T)	**0.834**	**0.941**	**0.947**	**0.864**	**0.850**	**0.766**	**0.831**	**0.878**	**0.756**	**0.877**
SIFT(KNN)	0.735	0.816	0.596	0.684	0.659	0.704	0.561	0.709	0.616	0.732
SIFT(V)	0.886	0.952	0.936	0.857	0.873	0.809	0.799	0.889	0.768	0.874
SIFT(T)	0.837	0.905	0.903	0.827	0.823	0.759	0.742	0.818	0.733	0.826
SIFT(V+T)	**0.889**	**0.953**	**0.937**	**0.861**	**0.877**	**0.812**	**0.805**	**0.896**	**0.776**	**0.897**
Color(KNN)	0.575	0.777	0.686	0.703	0.770	0.626	0.601	0.752	0.574	0.793
Color(V)	0.702	0.840	**0.843**	0.754	0.826	0.721	0.727	**0.864**	**0.703**	0.828
Color(T)	0.666	0.809	0.784	0.740	0.791	0.676	0.691	0.777	0.668	0.834
Color(V+T)	**0.715**	**0.853**	0.835	**0.782**	**0.850**	**0.726**	**0.754**	0.861	0.690	**0.865**
Unified(KNN)	0.794	0.883	0.841	0.794	0.850	0.720	0.695	0.852	0.630	0.866
Unified(V)	0.851	0.948	0.936	**0.885**	0.912	**0.822**	0.883	0.919	**0.800**	0.910
Unified(T)	0.873	0.924	0.933	0.826	0.877	0.788	0.826	0.901	0.785	0.873
Unified(V+T)	**0.901**	**0.959**	**0.944**	**0.885**	**0.922**	0.817	**0.890**	**0.931**	0.773	**0.923**
Combination(V)	0.891	**0.966**	0.953	0.902	0.918	0.823	**0.892**	0.933	0.816	0.917
Combination(T)	0.908	0.965	0.957	0.899	0.916	0.821	0.874	0.929	0.788	0.926
Combination(V+T)	**0.910**	0.965	**0.959**	**0.908**	**0.919**	**0.827**	0.887	**0.938**	**0.824**	**0.930**

Table 3.2: The AP values with different settings on PASCAL 2007 for each object category. Take the Gist feature as an example: "Gist(V)" denotes the result with the visual classifier; "Gist(T)" denotes the result with the text classifier; "Gist(T+V)" denotes the result by combining the text and visual classifiers. The performance of the text features depends on the strength of the visual features. "Unified(T)" usually works best among all the text classifiers and "Color(T)" usually works worst. We get better performance consistently by combining the text and visual classifiers. The results by combining all the text classifiers, all the visual classifiers and all the text and visual classifiers, are indicated by "Combination(T)," "Combination(V)" and "Combination(T+V)" respectively.

	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
Gist(V)	0.575	0.253	0.324	0.512	0.122	0.330	0.561	0.269	0.380	0.121
Gist(T)	0.520	0.207	0.296	0.509	0.089	0.335	0.509	0.227	0.302	0.178
Gist(V+T)	**0.580**	**0.272**	**0.362**	**0.548**	**0.189**	**0.392**	**0.578**	**0.295**	**0.383**	**0.203**
Grad(V)	0.571	0.230	0.238	0.403	0.116	0.333	0.551	0.308	0.397	0.184
Grad(T)	0.548	0.208	0.217	0.352	0.074	0.365	0.554	0.243	0.325	0.169
Grad(V+T)	**0.604**	**0.272**	**0.276**	**0.437**	**0.140**	**0.404**	**0.609**	**0.328**	**0.414**	**0.195**
SIFT(V)	0.510	0.297	0.249	0.412	0.122	**0.243**	0.416	0.330	0.324	0.212
SIFT(T)	0.288	0.254	0.237	0.367	0.104	0.184	0.309	0.320	0.264	0.209
SIFT(T+V)	**0.517**	**0.348**	**0.310**	**0.437**	**0.192**	0.241	**0.431**	**0.365**	**0.336**	**0.240**
Color(V)	0.367	0.124	0.220	0.215	0.112	0.085	0.323	0.134	0.242	0.075
Color(T)	0.400	0.084	0.215	0.215	0.078	0.107	0.332	0.112	0.154	0.098
Color(T+V)	**0.431**	**0.179**	**0.239**	**0.261**	**0.179**	**0.129**	**0.369**	**0.140**	**0.260**	**0.117**
Unified(V)	0.647	0.399	0.450	0.540	0.207	0.425	0.577	0.388	0.439	0.273
Unified(T)	0.580	0.349	0.407	0.545	0.120	0.329	0.565	0.366	0.352	0.170
Unified(V+T)	**0.666**	**0.445**	**0.512**	**0.580**	**0.232**	**0.450**	**0.619**	**0.438**	**0.459**	**0.295**
Combination(V)	0.675	0.407	0.423	0.581	0.239	0.432	0.646	0.421	0.449	0.279
Combination(T)	0.640	0.418	0.459	0.571	0.204	0.436	0.631	0.419	0.402	0.280
Combination(V+T)	**0.684**	**0.481**	**0.497**	**0.593**	**0.253**	**0.481**	**0.673**	**0.476**	**0.469**	**0.327**
	table	dog	horse	motorbike	person	plant	sheep	sofa	train	monitor
Gist(V)	0.289	0.270	**0.652**	0.364	0.679	**0.173**	0.167	0.281	0.541	0.316
Gist(T)	0.144	0.237	0.446	0.331	0.623	0.080	0.141	0.139	0.512	0.228
Gist(V+T)	**0.290**	**0.281**	**0.652**	**0.405**	**0.704**	0.130	**0.170**	**0.284**	**0.586**	**0.335**
Grad(V)	**0.356**	0.248	0.539	0.299	0.662	**0.118**	**0.131**	0.259	0.467	0.286
Grad(T)	0.205	0.179	0.432	0.251	0.601	0.081	0.080	0.171	0.409	0.207
Grad(V+T)	0.316	**0.253**	**0.575**	**0.336**	**0.670**	0.111	0.125	**0.263**	**0.485**	**0.332**
SIFT(V)	0.163	0.284	0.417	**0.243**	0.662	0.114	0.164	**0.196**	0.318	**0.227**
SIFT(T)	0.201	0.201	0.373	0.165	0.635	0.159	0.163	0.097	0.263	0.141
SIFT(T+V)	**0.239**	**0.321**	**0.474**	0.228	**0.687**	**0.182**	**0.255**	0.191	**0.339**	0.216
Color(V)	0.128	0.186	0.442	0.182	0.594	0.146	0.162	0.083	0.243	**0.122**
Color(T)	0.117	0.148	0.451	0.106	0.580	0.085	0.134	0.099	0.118	0.092
Color(T+V)	**0.195**	**0.220**	**0.513**	**0.192**	**0.615**	**0.148**	**0.163**	**0.121**	**0.255**	0.100
Unified(V)	0.373	0.343	0.657	0.489	0.749	0.330	0.324	0.323	0.619	0.322
Unified(T)	0.271	0.271	0.556	0.414	0.691	0.179	0.260	0.202	0.513	0.259
Unified(V+T)	**0.413**	**0.375**	**0.681**	**0.526**	**0.782**	**0.355**	**0.344**	**0.346**	**0.661**	**0.379**
Combination(V)	0.388	0.354	0.704	0.447	0.774	0.245	0.267	0.345	0.619	0.379
Combination(T)	0.336	0.335	0.648	0.484	0.738	0.233	0.305	0.252	0.612	0.295
Combination(V+T)	**0.442**	**0.392**	**0.715**	**0.528**	**0.786**	**0.272**	**0.322**	**0.350**	**0.665**	**0.402**

Table 3.3: The performance of the text features built with different numbers of Internet images on PASCAL 2006. We randomly select 200,000 and 600,000 images from the collection to construct the text features. The result is based on the average AUC values over the 10 object categories.

	200,000	600,000	1,000,000
Gist(T)	0.7116	0.8297	0.8370
SIFT(T)	0.6975	0.8104	0.8173
Grad(T)	0.7016	0.8093	0.8207
Color(T)	0.6496	0.7370	0.7436
Unified(T)	0.7413	0.8583	0.8606

Table 3.4: When we exclude category names and their plural inflections from the text features, there is little effect on the performance. We show results for Pascal 2006:
W - with category names; WO - without.

	bicycle	bus	car	cat	cow
W	**0.818**	0.915	0.932	**0.812**	0.843
WO	0.817	**0.917**	0.932	0.811	**0.848**
	dog	horse	motorbike	person	sheep
W	**0.744**	**0.820**	**0.878**	0.733	**0.875**
WO	0.738	0.816	0.876	**0.734**	**0.875**

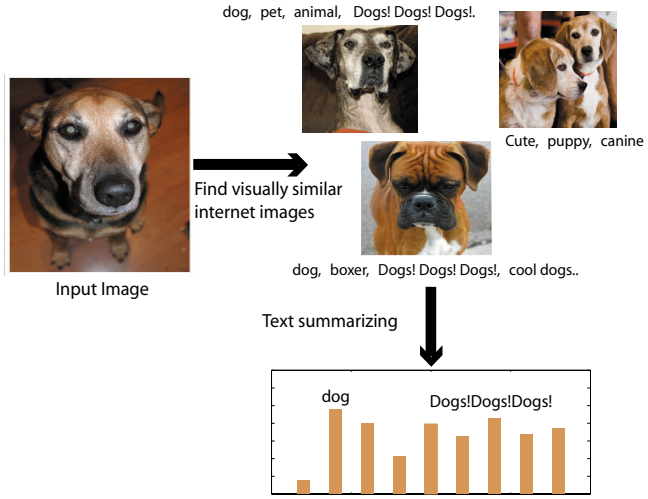

Figure 3.1: Illustration of our approach. For the input image, we find its similar Internet images (downloaded from Flickr). The text associated with these Internet images is summarized to build the text feature representation, which is a normalized histogram of text item counts. The Flickr text items can be tags such as "dog," and can be group names such as "Dogs!Dogs!Dogs!"

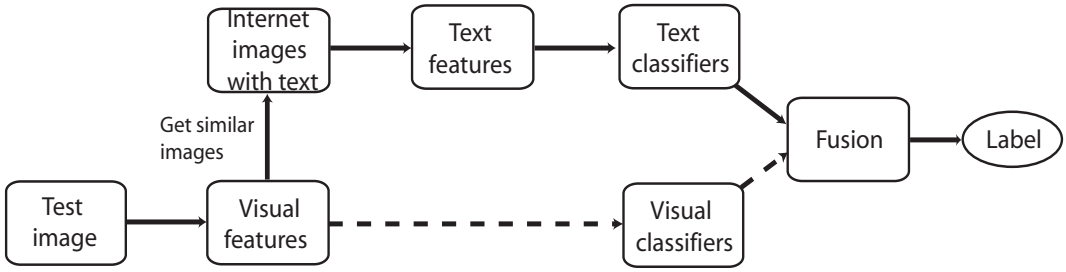

Figure 3.2: The framework of our approach. We have training and test images (here we only show the test image part). We also have an auxiliary dataset consisting of Internet images and associated text. For each test image, we extract its visual features and find the K most similar images from the Internet dataset. The text associated with these near neighbor Internet images is summarized to build the text features. Text classifiers which are trained with the same type of text features are applied to predict the object labels. We can also train visual classifiers with the visual features. The outputs from the two classifiers are fused to do the final classification.

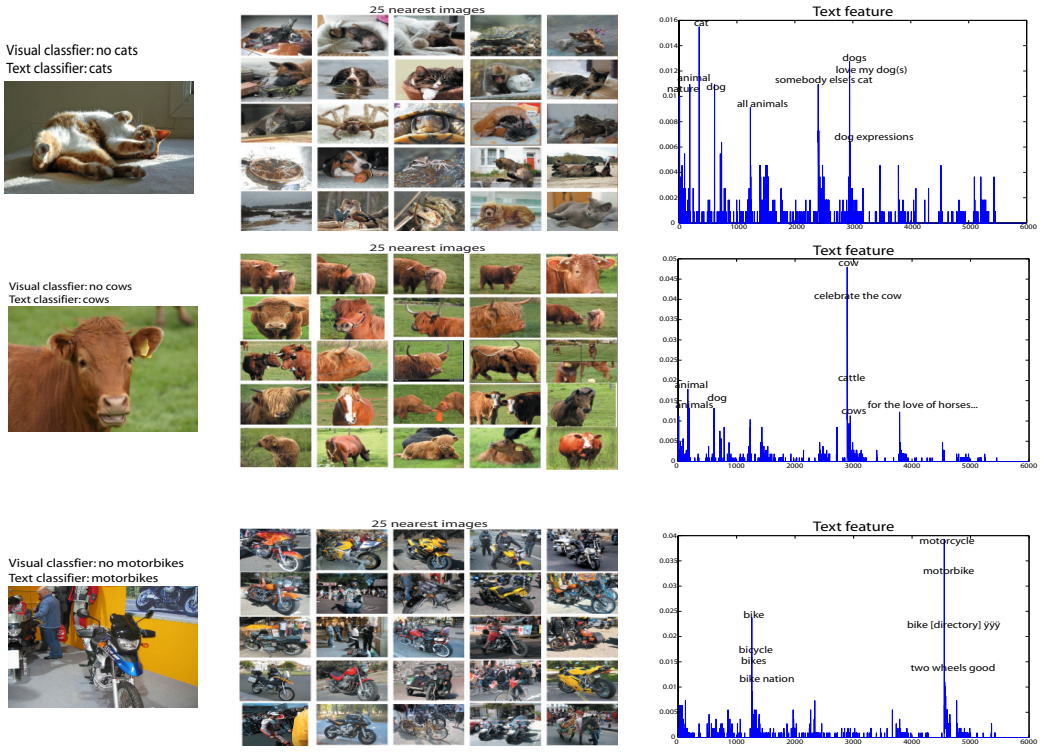

Figure 3.3: The left column shows the PASCAL 2006 images whose category labels cannot be predicted by the visual classifier, but can be predicted by the text classifier. The center column shows the 25 nearest neighbor images retrieved from the Internet dataset. The right column shows the built text feature vectors. In the first image, the cat is in a sleeping pose, which is unusual in the PASCAL training set. So the visual classifier gets it wrong. Some sleeping cat images are retrieved from the auxiliary dataset. Then the text features make a correct prediction.

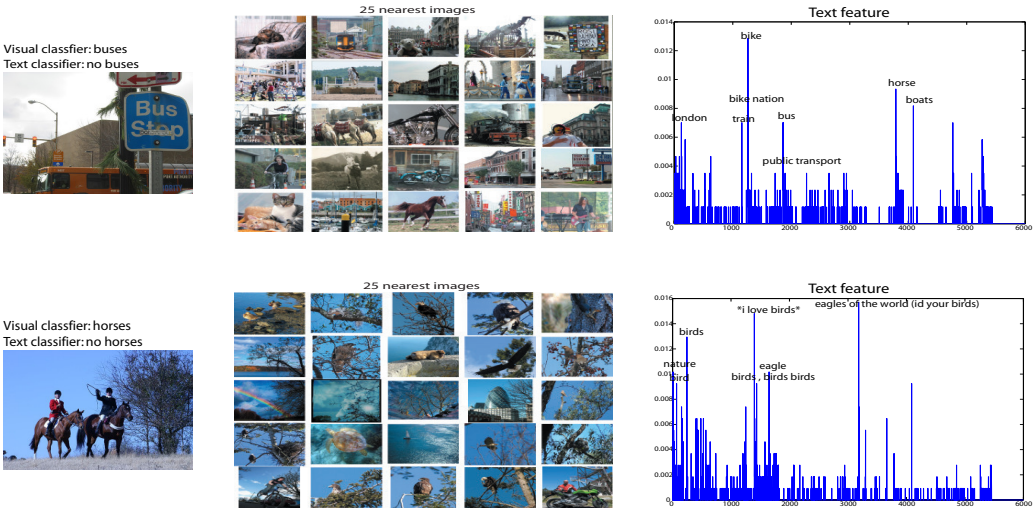

Figure 3.4: The left column shows the PASCAL images whose category labels cannot be predicted by the text classifier, but can be predicted by the visual classifier. The center column shows the 25 nearest neighbor images retrieved from the Internet dataset. The right column shows the built text features of the PASCAL images. The text features do not work here mainly because we fail to find good nearest neighbor images.

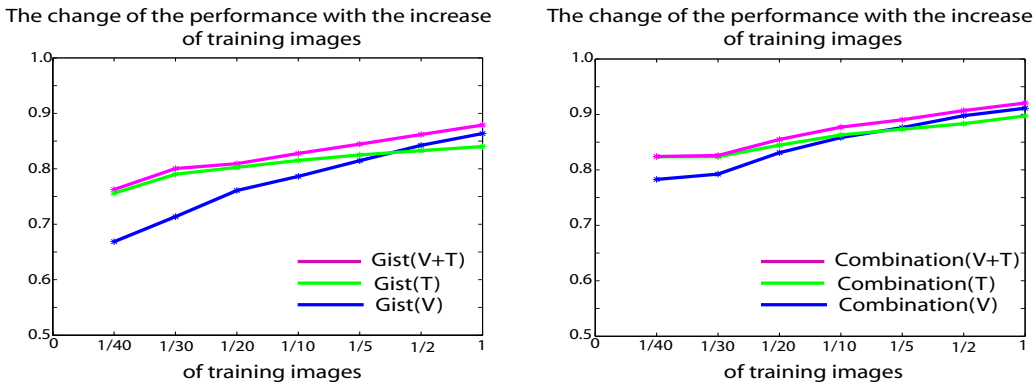

Figure 3.5: The performance with different numbers of training images on PASCAL 2006. We randomly select $1/40$, $1/30$, $1/20$, $1/10$, $1/5$, $1/2$ of the positive and negative images, respectively, in the training data for each category. The performance is shown by the average AUC values over all the categories. We do experiments by the "Gist" and the "Combination" of multiple classifiers. The text features outperform the visual features when there are only a small set of training images available. There is always improvement by combining the two types of features, but the gain is insignificant when the two classifiers are not comparable.

CHAPTER 4

FAST LEARNING WITH STOCHASTIC INTERSECTION KERNEL MACHINES

With more and more training data available, the computational cost of learning visual models becomes a big challenge. In this chapter, I will use image organization as a case study to introduce how we deal with large scale training data and how it is beneficial to computer vision applications.

Nowadays, digital cameras have made it much easier to take photos, but organizing those photos is still difficult. As a result, many people have thousands of photos sitting on their hard disk in some miscellaneous folders, but do not know how or have the time to organize them. Fortunately, the same digital explosion that created the problem may also supply the solution.

Using online photo sharing sites, such as Flickr, people have organized many millions of photos into hundreds of thousands of semantically themed groups. These groups show how people intend to determine similarity of images. Our idea is to transfer such knowledge to measure the similarity of two test images by learning from Flickr groups. Simply put, two images are similar in some sense if they are likely to belong to the same groups. If we can learn these group membership likelihoods, we can help a user sort through his photo collection by text or image-based query and refine the search with simple feedback. In doing so, we allow flexible, on-the-fly organization of his photo album.

But how can we learn whether a photo is likely to belong to a particular Flickr group? There are usually a lot of example images for each Flickr group. We can easily

download thousands of images belonging to the group and many more that do not, and train a discriminative classifier. However, the time that it would take to learn hundreds of categories is daunting, especially when we have many training examples for each category. In this chapter, we propose a new method to learn support vector machines (SVMs) with a Histogram Intersection Kernel (HIK), which has been proven to be successful in the image classification problem [14, 19]. We combine the kernelized stochastic learning algorithm from [98] with the support vector approximation method [99] proposed for fast classification. The result is an algorithm that is much faster and more accurate than the original stochastic learning algorithm, allowing us to learn from 5000 examples with 3000 features in just 15 seconds. This algorithm is also memory-efficient. It can process training examples one by one in a sequence. At each time, we only need to read one training example to the memory. This algorithm is useful for a wide variety of problems such as image classification and object detection.

Using the proposed fast training algorithm, we train classifiers to predict whether an image is likely to belong to a Flickr group. The set of prediction values is used to measure similarity. A simple Euclidean distance between the SVM outputs is found to work well. There are many Flickr groups. Some groups may capture more important visual properties than the others when used to determine image similarity. We adopt a metric learning algorithm to learn group weights, which are used to weight the prediction values when calculating the distance. The weights are learned with a big number of image triplets; each consists of two pairs, one of which has greater similarity than the other. In the learned metric, the distance between two similar images should be smaller than the distance between two dissimilar images. We learn metric parameters (Flickr group weights) by enforcing such constraints in a maximum margin framework.

Using the learned image similarity, we perform experiments on the Corel dataset *which was not obtained from Flickr*. We observe that learned image similarity outperforms the low level visual feature based similarity on the image matching task. Rele-

vance feedback [100] is an effective way to improve retrieval quality. With relatively little feedback, using learned Flickr similarity can boost the performance significantly, while using the low level visual feature based similarity cannot. Each Flickr group can be described by some key words, so our method also allows text based queries, which are also tested on the Corel dataset. Beyond matching and retrieval, we also apply our method to calculate kernel values for the image classification task on the PASCAL VOC datasets [39]. This new kernel outperforms the visual features based kernel on 12 object categories (20 object categories in total).

Related work

Measuring image similarity is a central topic of computer vision. How to represent images and how to calculate the distance between abstract representations are the two intrinsic problems of measuring image similarity. Popular image representations include color features, texture features [101–103], shape features [43, 104], and gradient features [2, 13]. Many features, such as the famous SIFT feature [2], are extracted locally. We need to concatenate all of the local features to form a global image representation. One popular way is the "bag of words" representation [50, 51]. It is later proven that enforcing coarse geometric structure such as the pyramid structure [14, 105] could further boost the performance in recognition and matching. In a "bag of words" representation, we need to construct a dictionary, which is traditionally generated using unsupervised ways such as k-means. Recently, learning the dictionaries to enforce sparse representation has become a hot topic [106]. When the representation is a single vector, the Euclidean distance or the chi-square distance can be applied to calculate the distance between two vectors. When the representation remains as a set of vectors, we can use the EMD distance [107]. We address the similarity measuring problem by learning image representation from Flickr groups. The most relevant work is [108]. We both advocate the use of features composed of category predictions for image matching. Our key observation, which differs from [108], is that Flickr provides an organizational

structure with thousands of categories that reflect how people like to group images, each with tens of thousands of examples, and our SIKMA classifier allows efficient and accurate learning of these categories.

Measuring image similarity has broad applications in computer vision. The significant one is image matching and retrieval. Space does not allow a comprehensive reciew of the enormous literature. Interested readers are encouraged to read the survey papers [109, 110]. Other applications include recognition by association [111], where one recognizes images by transferring class labels from found similar images. Another application is classification [19,51], where measuring image similarity is a critical step. Duplicate image detection is useful for detecting plagiarism [112].

We aim to exploit our image similarity learning algorithm to organize images on the fly. There is an extensive content-based image management literature, with recent reviews in [110,113]. Appearance [114] and iconic [115] matching are well established techniques. Clustering images as a way to expose the structure of a collection dates to at least [76, 116]. Relevance feedback has been used at least since [117]. Annotating images with words to allow word searches dates to at least [116]. None of these technologies works terribly well. Generally, users are querying for specific objects or object classes [118], and supporting object semantics is difficult. In recent work, Frome et al. [119] show that local metrics around examples built using Caltech 101 images give good retrieval behavior. Face annotation is especially interesting for image organization [120, 121]. Automatic image annotation and retrieval is hard. Active human annotation and labeling is usually needed to better annotate and organize personal images. [122, 123] exploit approaches to reduce the efforts of human intervention.

We wish to train a very large scale kernel SVM. In computer vision, bigger and bigger datasets are available such as the Lableme dataset [26] and the Imagenet [30]. There is a good survey of current results in large-scale kernel machine training in [124]. Algorithms are generally of two classes; either one exploits the sparseness of the Lagrange

multipliers (like SMO [125] and variants), or one uses stochastic gradient descent on the primal problem. Stochastic gradient descent has the advantage that, at each iteration, the gradient is calculated for only a single training example. Very good results can be obtained without touching every example [126, 127]. Kivinen et al. describe a method that applies to kernel machines [98]. We use a similar form of incremental training, exploiting a method of Maji et al. [99] for very quickly evaluating a histogram intersection kernel. This method is further extended to minimize an objective function enforcing similarity constraints [128]. Unlike [98], we do not need to drop any support vectors and maintain high efficiency. Reference [129] uses stochastic gradient descent to learn an additive classifier with a max margin criterion, avoiding the need to store any support vectors. With certain regularization, this is an approximation to the histogram intersection kernel SVM.

We use metric learning techniques to learn weights of Flickr groups in image similarity measuring. Distance metrics can be learned in an unsupervised or supervised fashion. Unsupervised methods include principal component analysis (PCA) [130], multidimensional scaling (MDS) [131] and locally linear embedding (LLE) [132]. Recently, supervised metric learning has drawn more attention [133,134], where "similar" or "dissimilar" information is given for a number of data points, and the learned metric must satisfy such constraints. In computer vision, metric learning has been adopted for object recognition [119] and data association [111].

4.1 Approach

We measure image similarity by calculating how likely they are to belong to the same Flickr groups. To do this, we first download thousands of images from many Flickr groups, which cover a wide range of common topics. For each group, we train a kernel machine classifier, which is used to predict the group membership of a test image. To get reliable classifiers, many (tens of thousands of) training images are used for each

group as the training data. Previous training methods usually cannot handle so many training images. In this chapter, we propose a new approach to train kernel machines using a histogram intersection kernel, namely Stochastic Intersection Kernel Machines (SIKMA). This algorithm is fast and memory efficient. There are many Flickr groups; we need to know which groups are more important when used to determine image similarity. We do this by adopting metric learning: given image triplets where one pair of images is more similar than another pair, we learn weights of Flickr groups to enforce similarity constraints. The learned weights weight different groups differently according to their importance. For two test images, we use the group classifiers to predict their group memberships. Image similarity is then measured using the prediction scores and the learned weights.

We introduce the details of each step in the following sections.

4.1.1 Downloading Flickr image groups

We download 103 Flickr image groups for our experiments, which capture a range of common topics. We choose Flickr groups that have many images and are informative. Some large Flickr groups (e.g., "10 million photos") are ignored because their images are diverse in topics. Groups that we use include objects, such as "aquariums" and "cars"; scenes, such as "sunsets" and "urban"; and abstract concepts, such as "Christmas" and "smiles".

Note that we could potentially make use of thousands of categories, each containing thousands of photographs.

Some Flickr groups have millions of images, and we cannot use all of them. For each group, we download around 15,000$\sim$30,000 images. Some examples from downloaded Flickr groups are shown in Figure 4.1.

4.1.2 Training group classifier using Stochastic Intersection Kernel Machines (SIKMA)

We train a histogram intersection kernel classifier for each Flickr group, which is used to predict group memberships of test images. Each Flickr group usually contains tens of thousands of images. To train a discriminative classifier, we also sample a large number of negative images from other groups. We usually use around 80,000 training images to learn a group classifier. Traditional approaches such as SMO cannot handle so many training examples because of the expensive computational and memory cost. In this chapter, we describe a new training algorithm called Stochastic Intersection Kernel Machines (SIKMA). It is an online learning algorithm, meaning that it processes training examples one by one in a sequence and does not have the memory issue. Using the histogram intersection kernel, we can combine the fast evaluation method developed in Maji et al. [99] with the stochastic gradient descent method, which leads to a very fast training algorithm.

We start by introducing the general stochastic kernel machines framework described in [98]. We have a list of training examples $\{(x_t, y_t), t = 1, \cdots, T, y_t \in \{-1, +1\}\}$. We aim to learn a decision function f: $X \longrightarrow R$, using a kernel machine. This yields $f = \sum_{i=1}^{N} \alpha_i K(x_i, \bullet)$ where K is a kernel function. Then for a test example u, the classification score is $f(u) = \sum_{i=1}^{N} \alpha_i K(x_i, u)$. In a primal method, we learn the kernel machines by minimizing the regularized empirical risk

$$L = \frac{1}{T} \sum_{t=1}^{T} l(f(x_t), y_t) + \frac{\lambda}{2} \|f\|^2 \tag{4.1}$$

where l is a loss function; a hinge-loss $l(f(x_t), y_t) = \max(0, 1 - y_t f(x_t))$ is used in the support vector machine framework, but some other loss functions such as log-loss can also be applied. In this approach part, we will introduce our approach based on the hinge-loss. Formula (4.1) can be minimized directly using the gradient descent method

in the primal space. At the tth iteration, we update f using:

$$f_t = f_{t-1} - \eta_t \frac{\partial L}{\partial f}|_{f=f_{t-1}} \tag{4.2}$$

η_t is the learning rate at the tth step. Doing (4.2) involves calculating

$$\sum_{t=1}^{T} \frac{\partial l(f(x_t), y_t)}{\partial f}|_{f=f_{t-1}} \tag{4.3}$$

which is very expensive when T is big.

Using the stochastic gradient method, we approximate the gradient by replacing the sum over all examples with a sum over some subset, chosen at random, and then take a step. It is usual to consider a single example. In Kivinen et al. [98], one sees this as presenting the training examples to the classifier in some random order, one by one, then updating the classifier at each example to get a set of f, $\{f_0, f_1, ..., f_T\}$, where f_0 is some initial hypothesis, f_{t-1} is the learned classifier by seeing the first $t-1$ training examples. Now assume we have f_{t-1}. When the tth training example comes, the new objective function is

$$Q = l(f(x_t), y_t) + \frac{\lambda}{2}\|f\|^2 \tag{4.4}$$

we update f as

$$f_t = f_{t-1} - \eta_t \frac{\partial Q}{\partial f}|_{f=f_{t-1}} \tag{4.5}$$

When l is the hinge-loss,

$$\frac{\partial Q}{\partial f}|_{f=f_{t-1}} = \frac{\partial l(f(x_t), y_t)}{\partial f(x_t)} \frac{\partial f(x_t)}{\partial f}|_{f=f_{t-1}} + \lambda f|_{f=f_{t-1}} \tag{4.6}$$

$$= \frac{\partial l(f_{t-1}(x_t), y_t)}{\partial f_{t-1}(x_t)} K(x_t, \bullet) + \lambda f_{t-1} \tag{4.7}$$

$$= -\sigma_t K(x_t, \bullet) + \lambda f_{t-1} \tag{4.8}$$

by writing

$$\sigma_t = \begin{cases} 1 \text{ if } y_t f_{t-1}(x_t) < 1 \\ 0 \text{ otherwise} \end{cases} \tag{4.9}$$

Then (5.7) is equivalent to

$$f_t = (1 - \lambda\eta_t) f_{t-1} + \eta_t \sigma_t y_t K(x_t, \bullet) \tag{4.10}$$

This update can also be written in terms of the Lagrange multipliers for the examples seen to date. In particular, we can write $\alpha_i = (1 - \lambda\eta_t)\alpha_i$ for $i < t$ and $\alpha_t = \eta_t \sigma_t y_t$.

We can see that when there are a large number of support vectors (this would happen in large datasets), it is expensive to calculate $f_{t-1}(x_t)$ in (4.19) because it involves calculating kernel values for many pairs of points. The NORMA algorithm developed in [98] keeps a set of support vectors of fixed length by dropping the oldest ones. As we shall see in the experiment section, doing so comes at a considerable cost in accuracy.

When a histogram intersection kernel is used, we do not need to drop any support vectors and can still maintain the efficiency. The histogram intersection kernel has a strong performance record in computer vision [14, 19]. Recently, Maji et al. [99] show that the support vectors of an intersection kernel machine can be efficiently represented. We exploit this trick to train a fast stochastic intersection without dropping any support vectors.

Write f_{t-1} as $\sum_{i=1}^{N_{t-1}} \alpha_i K(x_i, \bullet)$, where K denotes the histogram intersection kernel; write D as the feature dimension. Then

$$f_{t-1}(x_t) = \sum_{i=1}^{N_{t-1}} \alpha_i \sum_{d=1}^{D} \min(x_i(d), x_t(d)) \tag{4.11}$$

$$= \sum_{d=1}^{D} \sum_{i=1}^{N_{t-1}} \alpha_i \min(x_i(d), x_t(d)) \tag{4.12}$$

At each dimension d, if we have the sorted values of $x_i(d)$ as $\overline{x_i}(d)$, with the corresponding $\overline{\alpha_i}$, then

$$\sum_{i=1}^{N_{t-1}} \alpha_i \min(x_i(d), x_t(d)) \tag{4.13}$$

$$= \sum_{l=1}^{r} \overline{\alpha_l x_l}(d) + x_t(d) \sum_{l=r+1}^{N_{t-1}} \overline{\alpha_l} \tag{4.14}$$

where $\overline{x_r}(d) \leq x_t(d) < \overline{x_{r+1}}(d)$. As [99], we use M piecewise linear segments to approximately calculate (4.14). Given that feature histograms are normalized, each element of the feature vectors falls in the range of [0 1]. We divide this range to M (not necessarily even) bins, and the starting value of each bin is recorded in vector P.

Notice that the terms of Equation 4.14 contain only partial sums of α, rather than the values. This means that the complexity of representing the kernel machine has to do with these partial sums, rather than the number of support vectors. We can store these sums in tables, and update them efficiently. In particular, we have two tables B_1 and B_2 with dimensions $M \times D$, where M is the number of bins and D is the feature dimension. $B_1(m, d)$ contains the value $\sum_i \alpha_i x_i(d)\sigma_i$, $\sigma_i = 1$ if $x_i(d) < P(m)$ and zero otherwise; $B_2(m, d)$ stores the value $\sum_i \alpha_i \sigma_i$, $\sigma_i = 1$ if $x_i(d) \geq P(m)$ and zero otherwise.

To evaluate the function for $x_t(d)$, we quantize $x_t(d)$ and look up in B_1 and B_2. The two values are interpolated to approximately calculate (4.14). Since the elements of the tables are linear in the Lagrange multipliers, updating the tables is straightforward. At the tth iteration both B_1 and B_2 are multiplied by $1 - \lambda\eta_t$. If σ_t (see (4.19)) is non-zero, the tables B_1 and B_2 are updated accordingly by adding x_t.

Computational Complexity: From the above description, we can see that the computational complexity to train SIKMA is $O(TMD)$, where T is the number of training examples that are touched, M is the quantization bin size, and D is the feature dimension.

4.1.3 Learning weights of Flickr groups

As mentioned above, many Flickr groups (in our case, 103 groups) are used to determine image similarity. There might be redundant groups and some groups may represent more important visual information. We need to assign weights to different groups for similarity measurement (hopefully, the redundant groups will have weights of 0). A

metric learning algorithm is adopted to automatically learn group weights.

We use an idea similar to [119]. We have a triplet of images $(\{I_{n^1}, I_{n^2}, I_{n^3}\}, n = 1, \cdots, N)$. We know I_{n^1} is more similar to I_{n^3} than I_{n^2} to I_{n^3}, which is obtained by manual labeling or using class labels of images. Each of these images is represented as Flickr group prediction features, whose dimensionality is M. Each dimension corresponds to the prediction value of a Flickr group. We calculate the distance between two images (e.g., I_{n^1} and I_{n^3}) as

$$\sum_{m=1}^{M} w_m d^m_{n^1 n^3} \tag{4.15}$$

where $d^m_{n^1 n^3}$ denotes the distance between the two images only at the m dimension of the Flickr prediction features. A Euclidean distance is used here. The term w_m is the weight of the mth group. We learn w by forcing the distance between I_{n^1} and I_{n^3} to be smaller than the distance between I_{n^2} and I_{n^3}, with a big margin. This leads to minimizing the following objective function:

$$\frac{1}{N}\sum_{n=1}^{N} L(w^T d_{n^2 n^3} - w^T d_{n^1 n^3}, 1) + \frac{\lambda}{2}\|w\|^2 \tag{4.16}$$

$d_{n^2 n^3}$ and $d_{n^1 n^3}$ are vectors containing distance values calculated at each dimension, between pairs of images. L is the hinge-loss, and w is the weight vector, whose elements we require to be non-negative.

We also minimize (4.16) using the stochastic gradient descent method. Suppose at the tth iteration, the tth image triplet is chosen. Then an objective function $\widehat{Q}$ for this triplet is

$$L(w^T d_{t^2 t^3} - w^T d_{t^1 t^3}, 1) + \frac{\lambda}{2}\|w\|^2 \tag{4.17}$$

The gradient $\frac{\partial \widehat{Q}}{\partial w}_{w=w_{t-1}}$ is calculated as

$$\sigma_t(-d_{t^2 t^3} + d_{t^1 t^3}) + w_{t-1} \tag{4.18}$$

where

$$\sigma_t = \begin{cases} 1 \text{ if } w_{t-1} \cdot d_{t^2t^3} - w_{t-1} \cdot d_{t^1t^3} < 1 \\ 0 \text{ otherwise} \end{cases} \tag{4.19}$$

And the update is

$$w_t = (1 - \lambda\eta_t)w_{t-1} + \eta_t\sigma_t(d_{t^2t^3} - d_{t^1t^3}) \tag{4.20}$$

where w is forced to be non-negative. Using the same idea of [119], at each iteration of the stochastic gradient descent, we set all negative values as zeros.

4.1.4 Measuring image similarity

For two test images, we use the trained Flickr group classifiers to classify them and get prediction scores. Then the distance between prediction vectors can be calculated according to (4.15) using the learned weights.

We could also directly use the same weight for each dimension instead of the learned weight w. And this procedure does not decrease the performance much as shown in the experiment section. Once computed, this similarity measure can be used to perform image-based queries or to cluster images. Since we have names (groups) attached to each prediction, we can also sometimes perform text-based queries (e.g., "get images likely to contain people dancing") and determine how two images are similar.

4.2 Features and implementation details

We use four types of features to represent images and train the SVM classifier. The SIFT feature [2] is popularly used for image matching [37, 135] and object recognition [16] . We use it to detect and describe local patches. We extract about 1,000 patches from each image. The SIFT features are quantized to 1,000 clusters and each patch is denoted as a cluster index. Each image is then represented as a normalized histogram of the cluster indices. The Gist feature has been proven to be very powerful in scene

categorization and retrieval [58, 136]. We represent each image as a 960 dimensions Gist descriptor. We extract Color features in the RGB space. We quantize each channel to 8 bins, then each pixel is represented as an integer value range from 1 to 512. Each image is represented as a 512-dimensional histogram by counting all the pixels. The histogram is normalized. We also extract a very simple Gradient feature, which can be considered as a global and coarse HOG feature [13]. We divide the image to 4*4 cells; at each cell, we quantize the gradient to 16 bins. The whole image is represented as a 256-dimensional vector.

For each Flickr group, we train four SVM classifiers, one for each of the above four features. We combine the outputs of these four classifiers to be a final prediction on a validation dataset. In the validation procedure, we first randomly generate 10,000 different combinations of weights, then choose the one that maximizes the performance on the validation dataset. The final prediction is used to measure image similarity.

To compare our results with conventional visual similarity, we use a Unified visual feature, obtained by concatenating the above four visual features. Each feature is associated with a weight. The weights are learned on a validation set, where we force the images from the same categories to be close and images from different categories to be far away. This is similar to the metric learning algorithm mentioned above, and similar methodology can be found in [137, 138]. This feature tends to outperform each separate feature, and so gives a fair appearance baseline.

Most groups contain 15,000$\sim$30,000 images. To train a discriminative group classifier, we also sample about 60,000 negative images from other groups. Training each SVM using our SIKMA algorithm takes about 150 seconds per classifier, which tends to have between 5,000 and 8,000 support vectors. This is remarkable, considering that standard batch training is infeasible and that the previously proposed online algorithm NORMA [98] would take at least 10 times longer to produce a much less accurate classifier.

4.3 Experiments

In Section 4.3.1, we compare our fast histogram intersection SVM training algorithm (SIKMA) to alternatives. We show that our method is much faster and more accurate than a recently proposed stochastic learning method [98]. Our method is nearly as accurate as batch training on small problems involving a few thousand examples and enables training with tens of thousands of examples. For example, our method can train on 80,000 examples in 150 seconds, while batch training requires several hundred seconds to train with 5,000 examples.

In Section 4.3.2, we evaluate the usefulness of our learned similarity measure in several ways. We show that our similarity measure allows much better image matching in the Corel dataset and improves more with feedback than similarity based on the original image features. We can also perform text-based searches on non-annotated images in some cases.

4.3.1 SIKMA training time and test accuracy

We compare batch training, an existing stochastic learning algorithm NORMA [98], our proposed algorithm SIKMA for training SVMs with the histogram intersection kernel and linear SVM on the PASCAL VOC 2007 dataset [39]. We also report average precision results for our 103 Flickr categories.

Classifier Comparison. The batch learning method is implemented in LIBSVM [139]. LIBSVM does not support histogram intersection kernel directly, so we pre-compute the kernel matrix with a Mex function and use LIBSVM to solve it. In NORMA, the learning rate is set to be $\frac{0.71}{\lambda\sqrt{t}}$, and it keeps 500 support vectors at most. We validate λ for each category. Following [127], in SIKMA, the learning rate is set to be $\frac{1}{\lambda(t+100)}$, and λ is set to be 0.00005. For this comparison, the number of quantization bins is set to be 50. We also compare with linear SVM implemented in LIBSVM [139], where the parameter C is also cross validated.

This experiment is conducted on the PASCAL VOC 2007 image classification task, which has 20 object classes. For each class, there are 5,011 training images and 4,952 test images. As features, we use a 3000-bin histogram of quantized SIFT codewords, a standard feature for this task. The average precision (AP) results, training time and test time of different methods are compared in Table 4.1. The time of calculating features is not included to report the training and test time. Our method achieves similar accuracy to batch training when running for 6 rounds and is more accurate than either NORMA (also with histogram intersection kernel) or batch training of linear SVMs. We were surprised that the online kernelized learner (NORMA) tends to underperform the batch linear SVM. This seems to be due to the approximation of discarding support vectors with low weights and due to difficulty in choosing the learning rate for NORMA, on which we spent considerable effort. By contrast, our method is insensitive to the learning rate and consistently outperforms the linear classifier.

Our algorithm is much faster than NORMA and the batch algorithm. For larger problems, the speedup over batch will increase dramatically, and NORMA will be forced to make larger approximations at great cost to classifier accuracy. The same is true for memory requirements, which would make standard batch training impossible for problems with tens of thousands of examples.

In summary, our SIKMA algorithm makes it easy to train SVMs with the histogram intersection kernel on large datasets. Recent work by Maji et al. [99] makes classification nearly as fast as for linear kernels (this enables our training method). Together, these works are important because histogram intersection kernels tend to provide more accurate classifiers for histogram-based features that are used in many computer vision problems.

SIKMA parameters. In SIKMA, we need to use piecewise linear segments to approximate the continuous values. The approximation accuracy is controlled by the number of quantization bins. In Table 4.2, we show the training time, test time, and

performance of SIKMA using different numbers of quantization bins (each training example is visited three times). Training time and test time go up when we use more levels since we need to update more parameters. The best performance is achieved using 50 bins. There is not much change using more bins.

Performance on Flickr Categories. In Figure 4.2, we show average precision for our 103 Flickr categories. These are trained using positive group images (most have 15,000∼30,000 positive images) as well as about 60,000 negative images sampled from other groups. Each group has 20,900 held out test times: 500 positive and 20,4000 negative sampled from other groups. The average AP over these categories is 0.433.

4.3.2 Evaluation of learned similarity measure

We measure image similarity by calculating the distance between Flickr prediction features (in Euclidean distance or with weights learned by metric learning described in Section 4.1.3). The quality of the similarity measure is the most important factor in automatic organization, and we evaluate it in several ways. We can rank images according to similarity (image-based query) or cluster a set of images. We can also find images that are likely to belong to particular groups or have certain tags. We can also often say how two images are similar, suggesting the possibility of more intuitive feedback mechanisms. The following experiments are performed on 38,000 images from the Corel dataset (except where noted). Each image has a CD label and a set of keyword annotations, which we treat as ground truth for matching. There are 100 images per CD, and roughly 3-5 keywords per image.

Image Matching. We compare the performance of our Euclidean distance based similarity, metric learning based similarity, and also visual features based similarity. There are 38,000 Corel images involved. 8,000 images are randomly selected for the metric learning. In the learning procedure, we randomly generate 20,000 triplets of images. In each triplet, two images are considered to be similar if they have at least two

overlapping keywords; another two images are considered to be dissimilar if they have no overlapping keywords.

Among the remaining 30,000 images, 500 images are randomly chosen as query images. For each query image, the other images are ranked using the three similarity measurement approaches mentioned above. Images that have the same CD label or at least one keyword in common are considered correct matches; others are incorrect matches. For each query image, we calculate the AP value. For the visual features based similarity, the averaged AP value over all the query images is 0.110; for the Euclidean distance based similarity, the value is 0.123; for the metric learning based similarity, the value is 0.124. We can see that using Flickr groups (either in Euclidean distance or learned metric) helps compared to the standard visual features based similarity. Metric learning does not help much here, perhaps because the Flickr groups used are diverse and do not depend much on each other.

Figures. 4.3 and 4.4 compare the nearest neighbor images of a query image given by two approaches (visual features based similarity and Euclidean distance based Flickr groups similarity). Images are sorted by similarity in descending order from left to right, top to bottom.

These results indicate that the learned similarity works best when queries are related to the learned Flickr categories, but also provides an advantage in out-of-sample cases. As more Flickr categories are learned, fewer queries will be out of sample. Note that 1,000 classifications per second can be performed after computing the features, so it is entirely feasible to use thousands of Flickr categories (downloading tens of millions of images is the main obstacle for training).

The number of Flickr groups. In this experiment, we investigate how the matching performance changes with the change of the number of the Flickr groups. At each time, we randomly select a subset of Flickr groups (for example, 10). Then in the matching procedure, only the prediction scores of these groups are used to determine

image similarity (in a Euclidean distance). We calculate the averaged AP value using the 500 query images mentioned above. There is much variation when different subsets are chosen. We repeat 15 times for each number. The mean and standard deviation values for different numbers of groups are shown in Figure 4.5.

Image Matching with Feedback. Using a subset of 10 CDs, we also investigate the effectiveness of simple relevance feedback. We compare the performance of our features with that of visual features in a relevance feedback task. Because users will provide very little feedback (we use 5 positive and 5 negative examples), a good simulation of this task is demanding. We use the same CDs as [140], which are chosen to provide an unambiguous ground truth: 1 (sunsets), 21 (race cars), 34 (flying airplanes), 130 (African animals), 153 (swimming), 161 (Egyptian ruins), 163 (birds and nests), 182 (trains), 276 (mountains and snow) and 384 (beaches). Images are considered to match only if they have the same CD label. We compute average AP over 25 randomly selected queries.

For both visual features and Flickr prediction features, we initialize the weights of feature dimensions as ones. To simulate feedback, after each query, we select the top five negative examples and five randomly chosen positive examples from among the top 50 ranked images and label them according to ground truth. We use this to train a weight vector on our distance function. This is a very simple metric learning procedure. With the feedback, we aim to minimize the following function:

$$\sum_{i}^{10} y_i w \bullet (x_q - x_i)^2 \tag{4.21}$$

subject to $w_d \geq 0, \sum_d w_d = 1$, where x_q is the feature representation of the query image, x_i is the feedback example, y_i is 1 if it is positive, and 0 otherwise. If we had very extensive feedback, we would have a good estimate of the cost function. With relatively little feedback, the model of cost applies only locally around the current values of w. For this reason, we take a single step down the gradient, then project to

the constraints. The scale of the step is chosen on a validation set of 20 queries, and then fixed.

The average AP values on these 25 query images with three rounds of feedback are compared in Figure. 4.6. Note that the similarity based on Flickr prediction features improves more with each round of feedback than with the visual features. Figure. 4.7 shows the nearest neighbor images without feedback and with the first round of feedback for a query image. The selected negative images are shown in red and selected positive images are shown in green.

Semantic similarity of image pairs. In Figure. 4.8, we show six pairs of similar Corel images (measured in Euclidean distance). The text shows the Flickr groups which both of the images are likely to belong to.

Unsupervised clustering results on Corel dataset. Using the same 10 CDs as listed above, we also compare results on clustering. We represent these images with our prediction features (classification scores) and visual feature respectively. We cluster these 1000 images to 15 clusters in an unsupervised way (k-means). Each cluster is labeled with the most common CD label in this cluster. Each image is labeled by the cluster label accordingly. The accuracy of Flickr prediction features is 0.559 and the accuracy of visual features is 0.503.

Text-based queries. Because each Flickr category can be described with several words, we can support text-based queries. When users input a word query, we can find the Flickr group whose description contains such words. The corresponding Flickr group classifier is then used to classify personal photos. The photos with high confidence are returned to users.

We test this on the Corel dataset with two queries "airplane" and "sunset." There are about 38,000 images in total, from which there are 840 "airplane" images and 409 "sunset" images. We rank the images according to the Flickr group classification score. We get an AP value 0.28 for "airplane" and 0.16 for "sunset." In the 100 top-ranked

images for “airplane,” there are 52 true positives; in the 100 top-ranked images for “sunset,” there are 26 true positives.

The Corel images which are most relevant to “sunset” and “airplane” are shown in Figure. 4.9 according to the classification scores.

Classification. We can also use our Flickr group predictions as features for classification. In Table 4.3, we compare our prediction features with visual features. As implemented in [138], for the visual features, we train a chi-square kernel machine with the unified features (chi-square kernel is the state-of-the-art for histogram based image classification). Our group predictions features are not histograms, so we have to use an RBF kernel instead. Table 4.3 shows that our features are usually more effective than the visual features that are used to train the Flickr classifiers. Exceptions are objects that are typically in the background, such as tables, chairs, and bottles.

4.4 Discussion

In this chapter, we developed an idea to learn image similarity from Flickr groups. The motivation is that Flickr groups show how people would like to cluster similar images. Then two query images can be considered similar if the are likely to belong to the same set of Flickr groups. We also proposed SIKMA, an algorithm to quickly train an SVM with the histogram intersection kernel using tens of thousands of training examples. We use SIKMA to train classifiers that predict Flickr group membership. Our experimental results provide strong evidence that such learned image similarity works better on many tasks such as image matching and unsupervised clustering than directly measuring similarity with visual features.

There are several reasons that lead to the success of learned image similarity. First, the prediction bases are discriminative, as they are trained using tens of thousands of positive and negative training images. In this procedure, interesting visual patterns that are beneficial for matching and classification are preserved, while the others are aban-

doned. By mapping a new query image to these bases, we can measure how strongly it responds to such visual patterns. Second, the Flickr prediction features are very compact. As in our implementation, they only have 103 dimensions, while the traditional visual features may have thousands of dimensions. With compact representation, relevance feedback can be more effective because it deals with fewer parameters. This is also proven in our experiments. Third, the prediction features are semantic, as they correspond to some common concepts such as objects and scenes. Humans judge the similarity largely from the concept perspective, such as whether the two images contain instances of the same concept. So relating features to concepts is beneficial to computational similarity measurement as well.

One question regarding this line of work is what Flickr groups (concepts, in a more general sense) we should use as the bases and whether we know they can make a complete base in the visual space. It is difficult to answer this question in a theoretical way. But our proposed approach provides a practical solution. We could download many Flickr groups with many training examples and have distinctive topics, and run metric learning algorithms to select out the useful groups. However, in our experiment, the computationally selected Flickr groups with weights do not significantly outperform the manually selected groups with equal weights.

4.5 Tables and Figures

Table 4.1: Average precision (AP), training time and test time of the four SVM training methods are compared on the PASCAL VOC 2007 image classification task. All the values are averaged over the 20 object categories. HIK denotes the histogram intersection kernel. SIKMA (3 rounds) denotes that the SIKMA algorithm visits each training example three times. SIKMA (6 rounds) denotes that each training example is visited six times.

	Linear	SIKMA (3 rounds)	SIKMA (6 rounds)	Batch (HIK)
AP	0.362	0.429	0.436	0.440
training time (seconds)	-	23.1	46.7	638.0
test time (seconds)	0.5	3.9	3.9	236.8

Table 4.2: The AP values, training time, and test time of SIKMA using different numbers of bins. All the numbers are averaged over the 20 object categories. Each training example is visited three times in this implementation.

bins	10	20	50	80	120	150	200
AP	0.401	0.410	0.429	0.426	0.427	0.424	0.425
training time (seconds)	9.9	12.8	23.1	25.3	35.0	38.1	54.2
test time (seconds)	2.7	3.2	3.9	4.3	4.6	4.8	5.1

Table 4.3: The AP value with Flickr prediction features and visual features on PASCAL 2007 classification for each object class.

	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
Visual	0.647	0.399	0.450	0.540	**0.207**	0.425	0.577	0.388	**0.439**	0.273
Prediction	**0.650**	**0.443**	**0.486**	**0.584**	0.178	**0.464**	**0.632**	**0.468**	0.422	**0.296**
	table	dog	horse	motorbike	person	plant	sheep	sofa	train	monitor
Visual	**0.373**	0.343	0.657	0.489	0.749	**0.330**	**0.324**	**0.323**	0.619	0.322
Prediction	0.208	**0.377**	**0.666**	**0.503**	**0.781**	0.272	0.321	0.268	**0.628**	**0.333**

Figure 4.1: Image examples from ten download groups. Each row corresponds to a group. These groups are: aquariums, cars, Christmas, sunset, skyscrapers, boat, bonsai, food, fireworks, and penguin.

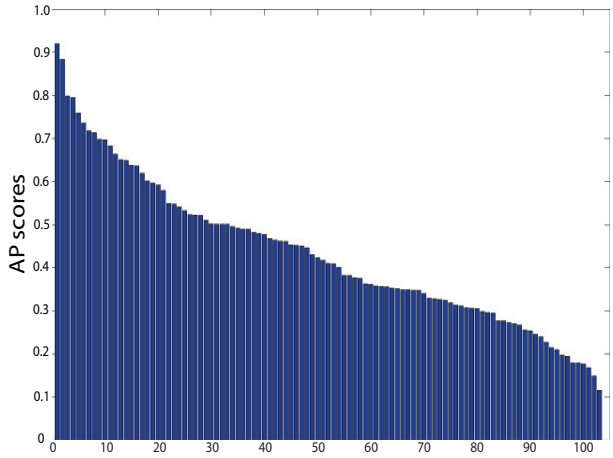

Figure 4.2: The AP scores of the 103 Flickr groups (categories). For each category, there are 20,900 held-out examples (500 positive). The five groups which get the highest AP values are laptop lunch, fireworks, pandas, socks and moon; the five groups which get the lowest AP values are love, art, trees, ice and light.

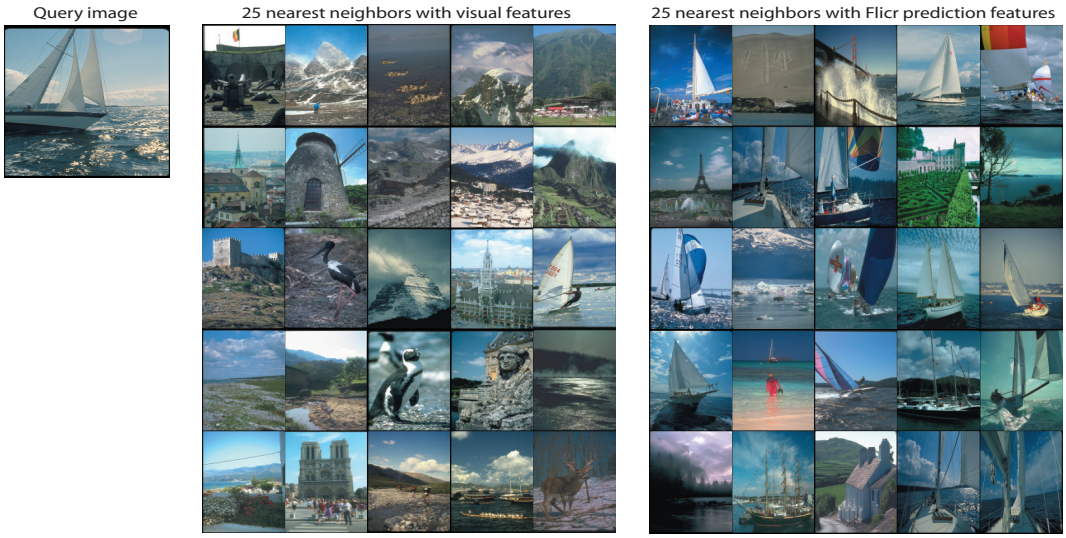

Figure 4.3: The left column shows a "ship" query image; the center column shows the 25 nearest neighbor images found with visual features; the right column shows the 25 nearest neighbor images found with Flickr features. The rank is from left to right, top to bottom.

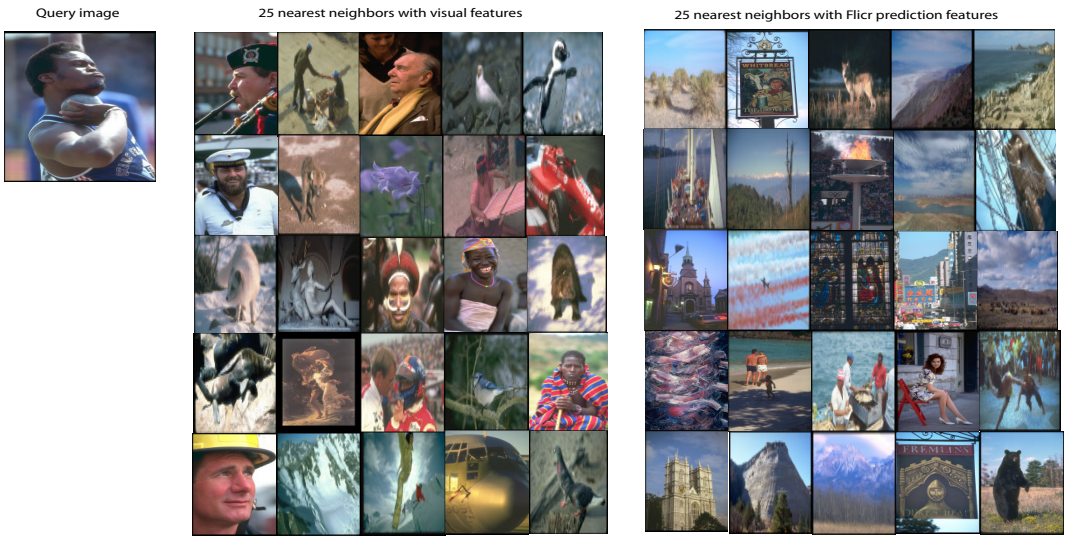

Figure 4.4: The left column shows a "person" query image; the center column shows the 25 nearest neighbor images found with visual features; the right column shows the 25 nearest neighbor images found with Flickr features. The rank is from left to right, top to bottom.

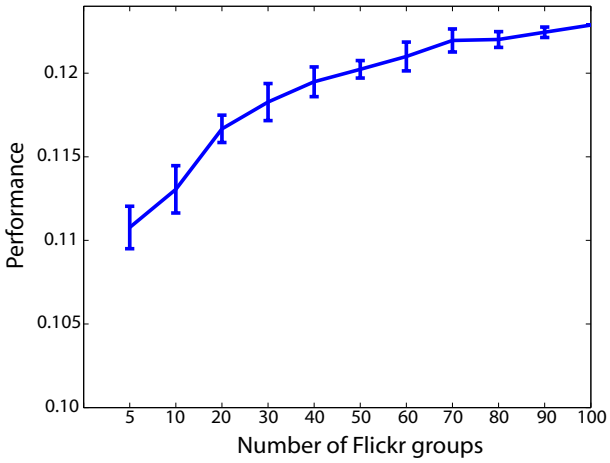

Figure 4.5: The AP values of using different numbers of Flickr groups. For each number, we repeat 15 times to randomly select a subset of groups. Both mean and standard deviation values are shown.

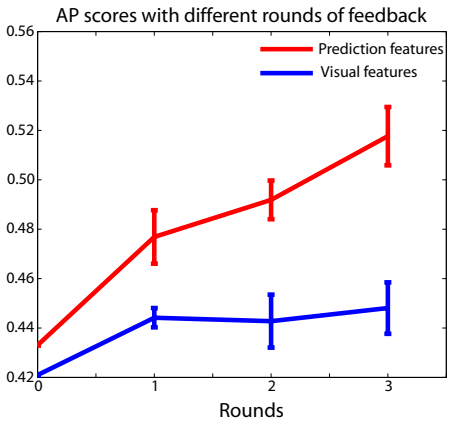

Figure 4.6: The average AP values with three rounds of feedback. The red line shows the results with Flickr prediction features and the blue line shows the results with visual features.

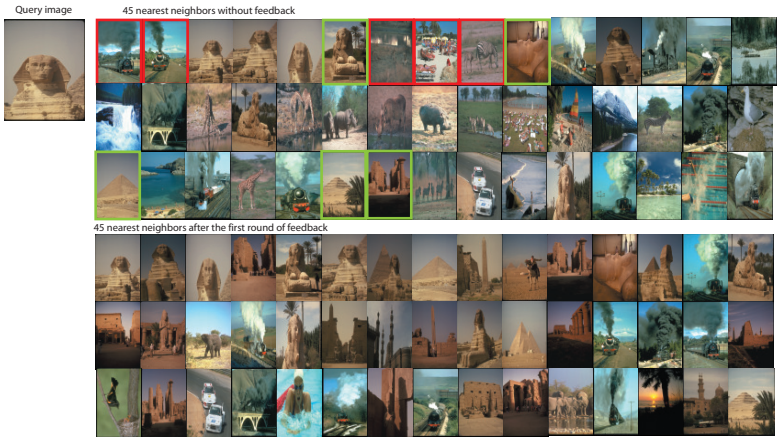

Figure 4.7: The left column shows the query image; the top row shows the 45 nearest neighbors found with the Flickr prediction features, with the five negative images (in red) and five positive images (in green) selected for feedback; after one round of feedback, we get the 45 nearest neighbors shown in the bottom row.

mountains(3.0) castles(1.2) sheep(1.2) turtles(0.9) cows(0.6)

sports(2.6) dances(2.0) weddings(1.0) toys(0.5) horses(0.5)

fireworks(15.6) Christmas(7.6) rain(4.0) water drops(2.5) candles(2.0)

Figure 4.8: Six pairs of similar Corel images. The text shows the top five Flickr groups which both of the images are likely to belong to. The value for each group in the parenthesis is $100 \times p(\text{group} \mid \text{image1})p(\text{group} \mid \text{image2})$.

Figure 4.9: The Corel images which are most relevant to the query "airplane," obtained by one-vs.-all classification with our SIKMA method, trained on the Flickr airplane group. Images are ranked according to their classifier score.

CHAPTER 5

LEARNING ROBUST OBJECT MODELS FROM FEW OR NO TRAINING EXAMPLES USING OBJECT SIMILARITY

There are very many object names. Training a system with many examples of each is likely to be difficult (most categories have few examples, as shown in Figure 5.1). Even if we could train such a system, doing so would not yield much insight into how people recognize objects. People seem to manage with few or no *visual* examples because there is much other information available to help identify objects. An important cue is being told what an object is "like." For example, few people know what a "serval" is, but when told it is like a leopard, but with longer legs and lighter body, most can identify one in a picture. "A serval is like a leopard" is a statement defining a new category in terms of existing categories.

Current methods to exploit similarity information in computer vision cannot deal with such statements. The usual method is metric learning. Here one measures similarity with some distance in a feature space, and adjusts feature weights to make objects more similar to those in the same category and dissimilar to those in different categories [119, 133, 134]; analogous procedures can be applied to measures of similarity that are not metric [141]. These methods cannot use explicit inter-category information. In the absence of category labels, data-dependent measures of smoothness can be used to weight features [142]. In each case, the result is a *global similarity procedure* — the metric is adjusted to be consistent with all available similarity information.

An alternative global similarity procedure uses multidimensional scaling (MDS) to obtain an embedding that is consistent with all similarity data. There is compelling

evidence that this is a poor model of human similarity judgments [143] (e.g., human judgments are not symmetric). Similarity judgments may not be consistent with one another or with new information (e.g. "a car is like a van," "a van is like a bus," and "a bus is like a train" do not mean that a "car is like a train"). MDS resolves this by seeking the best global embedding that is consistent with all statements. The method is also impractical for many categories, because we do not expect to have much detailed pairwise similarity information.

In this chapter, we exploit category similarity on a category by category basis. If we wish to learn a model of a "serval," we obtain a short list of similar categories from a human labeler; this would contain "leopard." We could just use these "leopard" instances as positive examples to train the model. This is not attractive, because the two categories are not the same. Worse, for many categories there may be nothing that is strongly similar. Our labeler marks "lamp" and "flower" as similar to "ceiling fan." These categories are similar enough to be helpful, but so different that we cannot mix them together.

Due to the uncertain degree of closeness between a target object class and its similar categories, we argue that labeled similar categories are just more similar to the target object than to other categories. For simplicity, we call these less similar categories "dissimilar categories" from now on. Our method uses similarity constraints as a form of regularization during learning. We require that the object model respond more strongly to examples of similar categories than to examples of dissimilar categories. For example, to learn a model of "serval," we obtain a few similar categories (e.g. "leopard") and some dissimilar categories (e.g. "grass" and "bird"). Then we require the model to respond more strongly to "leopard" than to "grass" and "bird." This process acts as a category dependent similarity regularizer.

We use this category dependent similarity regularization in a kernel machine framework [98]. It reduces the dimensions of the function space for learning. We show that

doing so leads to significant performance improvements in hard discrimination tasks, trained with very little data.

Objects might be similar in different ways; for example, a "zebra" is similar to a "horse" in shape, but similar to "crosswalk" in texture. For each aspect, we label similar and dissimilar categories and train a regularized kernel machine. Learned kernel machines are combined as the final output.

The following papers are relevant to our work. Fei-Fei et al. [65] exploit the tendency of object models to be similar to one another with a strong prior. Discriminative scores can provide more stable features, so fewer examples are required to learn a model [128]. There may be written descriptions of the object to be named, which can be exploited if the object is modeled in terms of *attributes*, properties of objects that can be observed and help describe them [144–148]. None of this work can exploit the comparative similarity of object categories.

5.1 Learning object models with comparative similarity

We assume we wish to learn a model to identify a named object. We have few or no positive training examples, many negative examples, and some categories identified as "similar" or "dissimilar." We first show how to incorporate this information into the training process for a kernel machine. We then show how to evaluate different aspects of similarity (e.g. similarity in color, in texture, and so on).

5.1.1 Incorporating comparative similarity into training

We aim to learn an object model F for each name. We follow [14, 51] and use a kernel machine. Write $F = \sum_{i=1}^{N} K(x_i, \bullet)$, where $K(x_i, \bullet)$ is a kernel basis function (we use a histogram intersection kernel). We have a set of T training examples in a feature representation $\{(x_t, y_t), t = 1, ..., T, y_t \in \{+1, -1\}\}$.

The usual procedure is to learn F by minimizing:

$$\frac{1}{T}\sum_{t=1}^{T} L(F(x_t), y_t) + \frac{\lambda}{2}\|F\|^2 \tag{5.1}$$

where $\frac{\lambda}{2}\|F\|^2$ is a regularization term, and L is the hinge loss $L(F(x_t), y_t) = \max(0, 1 - y_t F(x_t))$. However, accurate learning requires numerous positive examples (see Section 5.2).

We do have examples from categories that are similar to the name we seek to learn. A simple approach is to use these as positive instances; we use this approach as a baseline (see section 5.2). When we append them as positive instances, we weight them with a parameter in the similar form of Equation 5.3.

A good object model would respond strongly to whatever positive examples there happen to be, but would also respond more strongly to similar examples than to dissimilar examples. This lends the problem an ordinal character — our method should rank similar examples more highly than dissimilar examples, rather like an ordinal SVM [149]. Ordinal SVM attempts to learn a function $h(x)$ such that $h(x_i) > h(x_j)$ for any pair of examples where $\text{rank}(x_i) > \text{rank}(x_j)$. However, we do not have a full ranking of all examples, so we cannot use a conventional ordinal SVM.

Our model F should be positive for positive instances and negative for negative instances, and it should be larger for similar instances than for dissimilar instances. The first two requirements are straightforward to express with the hinge loss. For the third, if g_n^s is an instance from a similar category and g_n^d is an instance from a dissimilar category, $F(g_n^s)$ should always be larger than $F(g_n^d)$, with some margin. We impose this constraint by preparing a set of N similar-dissimilar pairs, and scoring $L(F(g_n^s) - F(g_n^d), 1)$, where L is the hinge loss. This acts as a regularization term. There could be very many pairs. If there are many positive examples, then the similarity constraint is less significant, but if there are few, it is an important constraint. This means the weight placed on this similarity term should depend on the number of positive examples T_p.

We must choose F to minimize

$$\frac{1}{T}\sum_{t=1}^{T} L(F(x_t), y_t) + \alpha(T_p)\frac{1}{N}\sum_{n=1}^{N} L(F(g_n^s) - F(g_n^d), 1) + \frac{\lambda}{2}\|F\|^2 \tag{5.2}$$

where $\alpha(T_p)$ represents the weight on similarity as a function of the number of positive training examples. The first term is the empirical loss of the target category, the second term imposes similarity constraints (it is also considered to be a regularization term), and the third is the conventional regularizer. Generally, if there are few positive examples, $\alpha(T_p)$ should be large, and if there are many, it should be small. We use

$$\alpha(T_p) = \frac{A}{1 + e^{B(T_p - C)}} \tag{5.3}$$

where A, B and C are parameters chosen by a cross validation procedure applied to set of categories, which have multiple training instances. These parameters are fixed for all other categories. Training is by stochastic gradient descent; details are in Section 5.1.3. The resulting F is a ranking function.

5.1.2 Learning with aspect based similarity

Objects might be similar in different ways, which are called aspects in this chapter. Using aspect based similarity could: (1) help human labelers to label similar categories; (2) help design object representation (e.g., if two objects are similar because of color, we should use color features to represent objects and learn F). We have three types of aspects: texture, shape and scene. We label different but similar categories and use different features for different aspects: SIFT words for texture, pyramid HOG for shape, and SIFT words on the image for scene. More details can be found in Section 5.2.2.

Using the idea described in the above section, we train a kernel machine for each aspect of similarity. Each produces a ranking function F_a for a particular aspect a. We can get better results by combining responses produced by multiple aspects based

similarity. We calibrate the responses, then sum them to obtain an overall ranking. Calibration is important, because the same degree of similarity might give different values of ranking function for different aspects.

We calibrate by applying a linear transformation to the ranker output so that the strongest (resp. weakest) response on a dataset is 1 (resp 0). The same dataset is used for each aspect, so that the same number of positives should be present in each case, but labels are not known. Again, no more complex procedure is possible, because we have few positive examples.

We linearly combine the calibrated classification responses. The combination weights are learned using validation on categories with multiple training instances.

5.1.3 Training

To train the models, we use a stochastic gradient descent method [98, 150]. Stochastic gradient descent selects some terms in the objective function at random and takes a small step in a direction that will minimize them. Our terms could be either loss at a labeled example, or loss for a similar-dissimilar pair. The algorithm becomes:

At the ith iteration, select a single loss term at random:

1. If this is loss at a labeled example x_t, then follow the procedure of [98, 150]. The update is

$$F_i = (1 - \lambda\eta_i)F_{i-1} + \frac{1}{T}\eta_i\sigma_i y_t K(x_t, \bullet) \tag{5.4}$$

where $\sigma_i = \begin{cases} 1 \text{ if } y_t F_{i-1}(x_t) < 1 \\ 0 \text{ otherwise} \end{cases}$

2. If we have a similar-dissimilar pair $\{g_n^s, g_n^d\}$, the term $\widehat{O}$ for that pair is

$$\frac{1}{N}\alpha(T_p)L(F(g_n^s) - F(g_n^d), 1) + \frac{\lambda}{2}\|F\|^2 \tag{5.5}$$

and the gradient $\frac{\partial \widehat{O}}{\partial F}|_{F=F_{i-1}}$ is

$$\frac{1}{N}\alpha(T_p)\sigma_i(K(g_n^d, \bullet) - K(g_n^s, \bullet)) + \lambda F_{i-1} \tag{5.6}$$

where $\sigma_i = \begin{cases} 1 \text{ if } F_{i-1}(g_n^s) - F_{i-1}(g_n^d) < 1 \\ 0 \text{ otherwise} \end{cases}$ and the update becomes

$$F_i = (1 - \lambda\eta_i)F_{i-1} + \frac{1}{N}\alpha\eta_i\sigma_i(K(g_n^s, \bullet) - K(g_n^d, \bullet)) \tag{5.7}$$

We use a histogram intersection kernel for K, allowing us to use the fast training algorithm of [150]. We modify their public training code to learn our models; an alternative is to use the method of [129].

5.2 Experiments

5.2.1 Procedures

Dataset: We use 2831 images from Labelme [26] as a test bed. Object regions are fully annotated within images. We manually reword the object names by correcting misspelled words, removing non-noun words (e.g., "a"), and passing to the most common nouns (e.g., replacing "pedestrian walking" with "person"). This leaves 972 object categories in total.

As Figure 5.1 shows, the distribution of object categories in our dataset is heavily long-tailed (which is also suspected to be true in the real world). Around 600 categories have less than 6 instances. Only 70 categories have more than 100 instances. We randomly select 1,500 images as training data and the other 1,331 images as test data.

General similarity annotation: We select 90 object categories, which have more than 60 instances, as prototype categories. We use 225 test categories, each of which has at least one test instance. For each category, a human volunteer identified up to five similar objects from these prototype categories without seeing any images from the dataset. In this labeling procedure, no extra instructions are given on which aspect

(e.g., shape or texture) should be used to judge similarity. So it is called "general similarity," in contrast to the "aspect based similarity" described below.

This work was checked by a second volunteer, who broke the similarity judgments into four cases: synonymous (e.g. category "beach rock" and prototype "rock"); near synonymous (e.g. "worktop"; "bar counter"); different (e.g. "bird"; "flag" — the labeler felt both flap in the sky); and very different (e.g. "ceiling fan";"flower"). More examples of similarity annotation are shown in Table 5.1.

Aspect based similarity: In addition to the general similarity labels for 225 categories, we also label aspect based similarity for 50 categories. As mentioned before, there are three types of aspects: texture, shape and scene. The similarity is also labeled without seeing any images in the dataset. Scene similarity is obtained by finding prototype objects which tend to appear in the same scene (scene types are mainly restricted to "indoor" or "outdoor") . All images containing instances of these prototype objects are of similar scenes.

5.2.2 Features and parameters

For general similarity and texture similarity, we represent objects as an 800-dimensional histogram of SIFT [2] words. For scene similarity, we also represent images as an 800-dimensional histogram of SIFT words. For shape-based similarity, we use the PHOG (for Pyramid of Histograms of Orientation Gradients) described in [151].

We use around 20,000 negative examples and 20,000 pairs of similar-dissimilar examples to train the models. When training the kernel machines, the learning rate η_i is set to be $\frac{1}{i+100}$, and λ is set to be 0.00005. It usually takes 50$\sim$120 seconds to train one object model.

When training one object model, all the other classes are used as negative. In the test procedure, we classify each test image region and output a classification score. AUC values are calculated for each class. Since our goal in this chapter is to investigate how

similarity helps to categorize uncommon objects rather than detect objects as in [152], we directly use the ground truth segmentations of test images to extract object regions. There are 21,803 test regions in total.

5.2.3 Baselines

We compare our algorithm with two baselines. Baseline 1 uses all available positive and negative examples in the usual way. If there are no positive examples, it outputs a random guess. Baseline 2 uses instances from similar categories as positive examples (weighted with a weight in the similar form of Equation 5.3). For aspect based similarity, each baseline is obtained as described for each aspect.

5.2.4 General similarity improves AUC

If we present a method with one positive and one negative example, the area under the receiver operating curve (AUC) gives the probability that the method will correctly identify them. AUC is a good measure of performance for a task like naming with few examples. Instead of using the standard AUC, we adopt a balanced AUC, where each test example is weighted by $\frac{1}{N}$ (N is the total number of test instances from the same category). This can better measure how well the learned models are against all the other categories rather than against some very common categories. Our general similarity method produces strong improvements in AUC for all test categories, especially when there are no positive examples (Figure 5.2). AP is a less helpful measure because there are very few positive examples and approximately 20,000 negative examples, so all scores are very small and unstable.

We also show some qualitative results in Figure 5.3. Our method gets better AUC values and ranks more sensible regions on the top.

Our method can reach very high AUC scores on many categories even if they have no training examples. Figure 5.4 shows the number of categories (from the 110 categories with no training examples) whose AUC values exceed a set of AUC thresholds.

More than 40 categories have bigger AUC values than 0.9.

This effect is not purely due to synonymy in the labels. We sort categories by the strongest similar prototype (strongest: synonymous to weakest: very different). Overall, there are 53 categories with at least one synonymous similar prototype; 70 categories with at least one near synonymous similar prototype and no stronger; 90 categories with at least one different similar prototype and no stronger; and 12 categories which have only very different similar prototype. We show average AUC scores for each type in Figure 5.5. We can see that even if the similar prototypes are at best "different" or "very different," using similarity yields a better AUC.

The number of similar classes for our examples ranges from one to three, with very few categories having more. We investigated the effect of the number of similar classes on the improvement in AUC, but found no effect. We believe that it is the quality, rather than the number, of similar categories of labeled similar categories that matters.

5.2.5 Aspect based similarity improves AUC

We test multi-aspect similarity on 50 categories (mainly indoor objects such as bookcase and dish towel). When combining classification scores from different aspects, we first calibrate them as introduced in Section 5.1.2. Then we weight them linearly. The weights are learned using the validation set on 5 categories, with at least 2 training examples. The learned weights are 0.554, 0.341 and 0.106, for texture aspect, shape aspect and scene aspect respectively. The results are shown in Table 5.2. Combining cues from different aspects helps. The scene cue is useful, especially for objects which only appear in specific scenes. If one object appears in very diverse scenes (such as "person"), the scene cue is not going to help.

Within these 50 categories, there are 27 categories which are also labeled with general similarity. The average AUCs of these 27 categories are compared in Table 5.3.

5.2.6 General similarity improves correspondence

An improvement in AUC is very helpful in finding correspondence between regions and weakly labeled object names [77].

We choose 197 images from the test set which have at least three regions from any of our 225 test categories. Their labels are weakly labeled, meaning that we do not know which label goes to which region. Our task is to use the learned models to establish the correspondence.

We solve for correspondence with a maximum weight bipartite matching (using the Hungarian algorithm [153]), where weights are given by calibrated classification scores. The matching results are region labels. We calculate matching accuracy for each class. The values are averaged for comparison to avoid effects of large categories (see Table 5.4).

In Figure 5.6, we show the average accuracy values on categories by the number of training instances. Our method yields a large improvement on categories with zero or few training instances. This is important because the distribution of objects in the real world is long-tailed. Correspondence examples are shown in Figure 5.7.

5.3 Tables and Figures

Table 5.1: Example categories and their general similarity annotation. The similarity type is labeled by a second volunteer.

object	similar categories	dissimilar categories	similarity type
shrub	flower, grass, hedge	bowl, vase, bottle, umbrella	synonymous
number	text	tv, sink, box, bush	synonymous
body	torso	grass, road, motorbike, hedge	nearly synonymous
picture frame	painting	towl, bed, floor, sofa, bush	nearly synonymous
plant pot	bowl	rug, sign, sand, door	different
bird house	box, book	flower, sofa, floor, motorbike	different
gloves	curtain, flag	grass, desk, door, bottle	very different
fireplace	fence, railing	step, path, bicycle, snow	very different

Table 5.2: Average AUC values on 50 categories for different aspect similarity and their combinations. “T” denotes the result using texture based similarity, “Sh” denotes the result using shape based similarity, “Sc” denotes the result using scene based similarity. Combining multiple aspects helps.

T	Sh	Sc	T+Sh+Sc
0.735	0.725	0.686	0.783

Table 5.3: Average AUC values on 27 categories for general similarity, different aspect based similarity and their combination. “G” denotes the result using general similarity. Combining multiple aspects works comparably well.

G	T	Sh	Sc	T+Sh+Sc
0.769	0.748	0.702	0.653	0.771

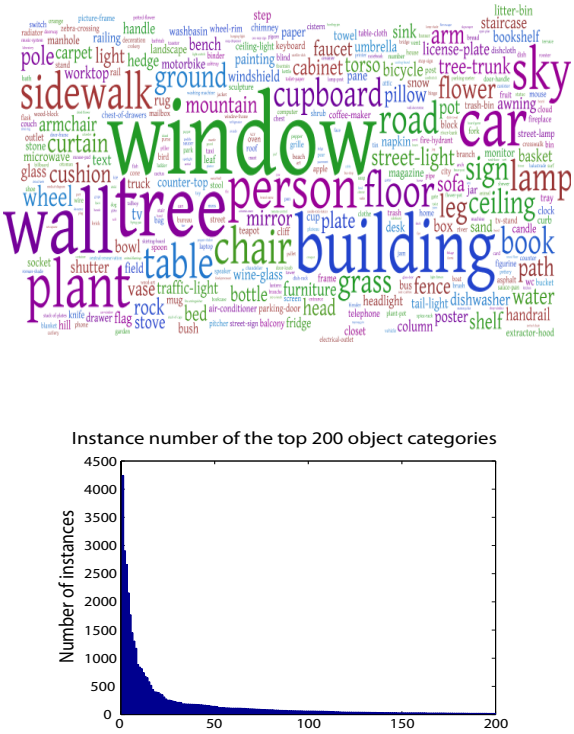

Figure 5.1: Most categories in the dataset we use (which is a part of Labelme) have few or no examples. The top image shows "object clouds." Objects with bigger names have more instances. Most objects have small names because they have few examples. The bottom image shows number of instances for the top 200 objects. The top 5 categories are window, tree, wall, building and car. The number of instances decays very quickly.

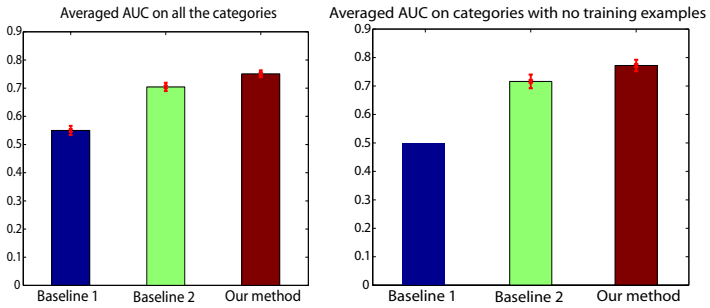

Figure 5.2: General similarity improves AUC, even when there are no examples. Left: AUC averaged over all the 225 test categories for baselines and for our method. Right: comparison for the 110 test categories which have *no* positive examples, both with standard error bars.

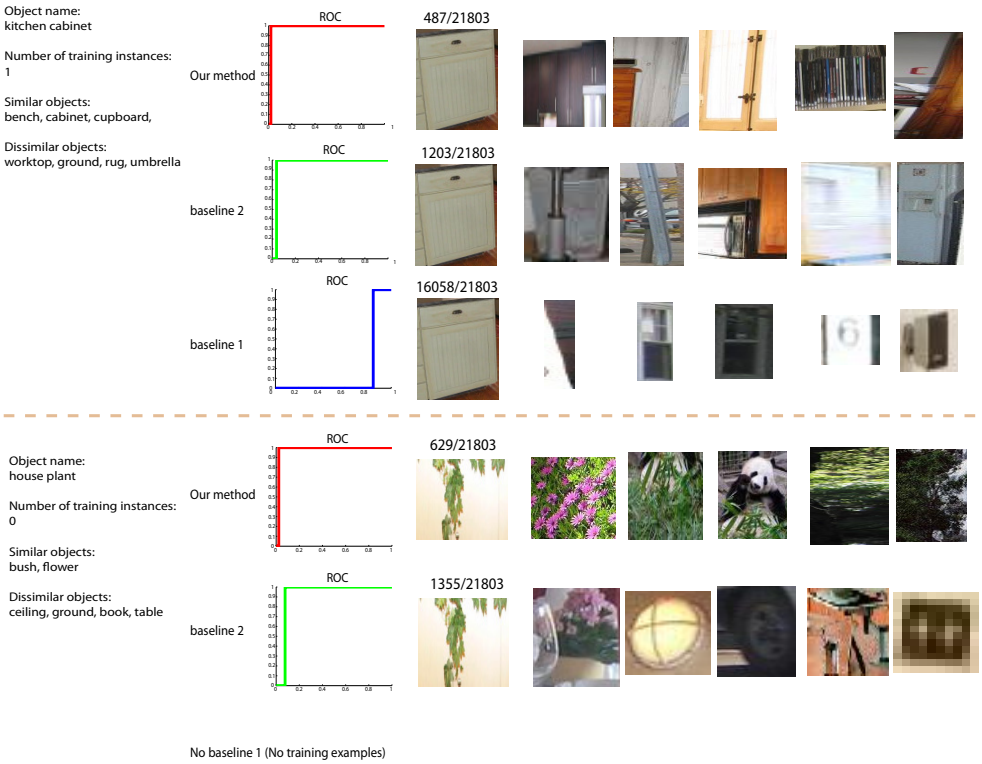

Figure 5.3: Classification results for two categories. For each category, the first row shows results using our method; the second row shows results using baseline 2; the third row shows results using baseline 1 (if there are positive training examples). In each row, the first figure shows the ROC curve; the second image shows the top-ranked positive test instance (for each of these two object classes, there is only one positive test instance), and the number above the image shows the rank (out of 21,803 test instances); the following five images show top-ranked test regions. Our method yields better AUC (rank) than baseline 2 and baseline 1. It also ranks more reasonable regions on the top.

Table 5.4: Average matching accuracy using classification scores by different methods. The accuracy is averaged over categories. Using our method can establish better correspondence.

Baseline 1	Baseline 2	Our method
0.440	0.486	0.535

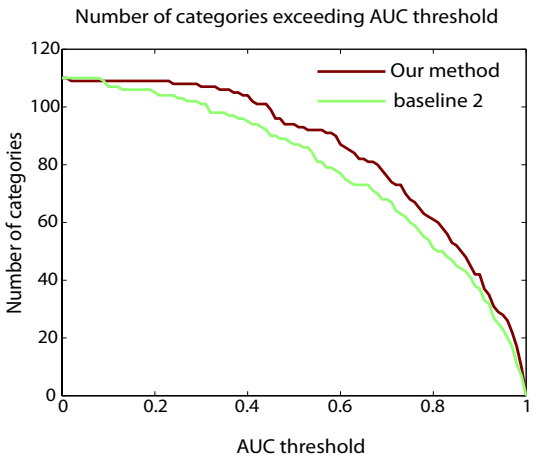

Figure 5.4: Number of categories for which AUC is greater than a threshold. These categories have no positive training examples. The x-axis denotes the AUC thresholds. The y-axis denotes the number of categories whose AUC values exceed the thresholds. There are more than 40 categories with bigger AUC values than 0.9 using our method. Our method consistently yields more categories than baseline 2 in the high AUC area. Note that some categories have AUC values smaller than 0.5, because the AUC is unstable when there are only a few positive test examples.

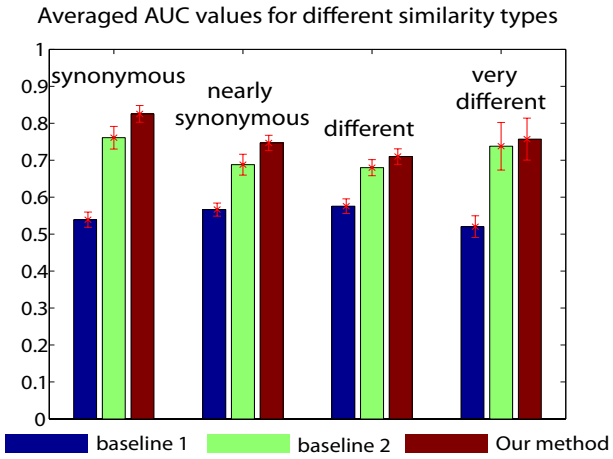

Figure 5.5: General similarity improves average AUC score; the effect is not due to synonymy. We show the average for all categories with at least one synonymous similar prototype; all with at least one near synonymous similar prototype and no stronger; all with at least one different similar prototype and no stronger; and all which have only very different similar prototype. Note there are across the board improvements, which are strong compared to error bars.

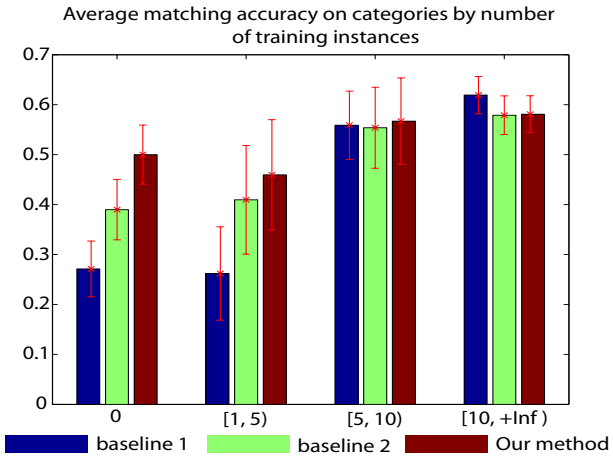

Figure 5.6: Average matching accuracy on categories by number of training instances. On the categories with zero or few training examples, using similarity helps a lot. Our method also gets better results than baseline 2.

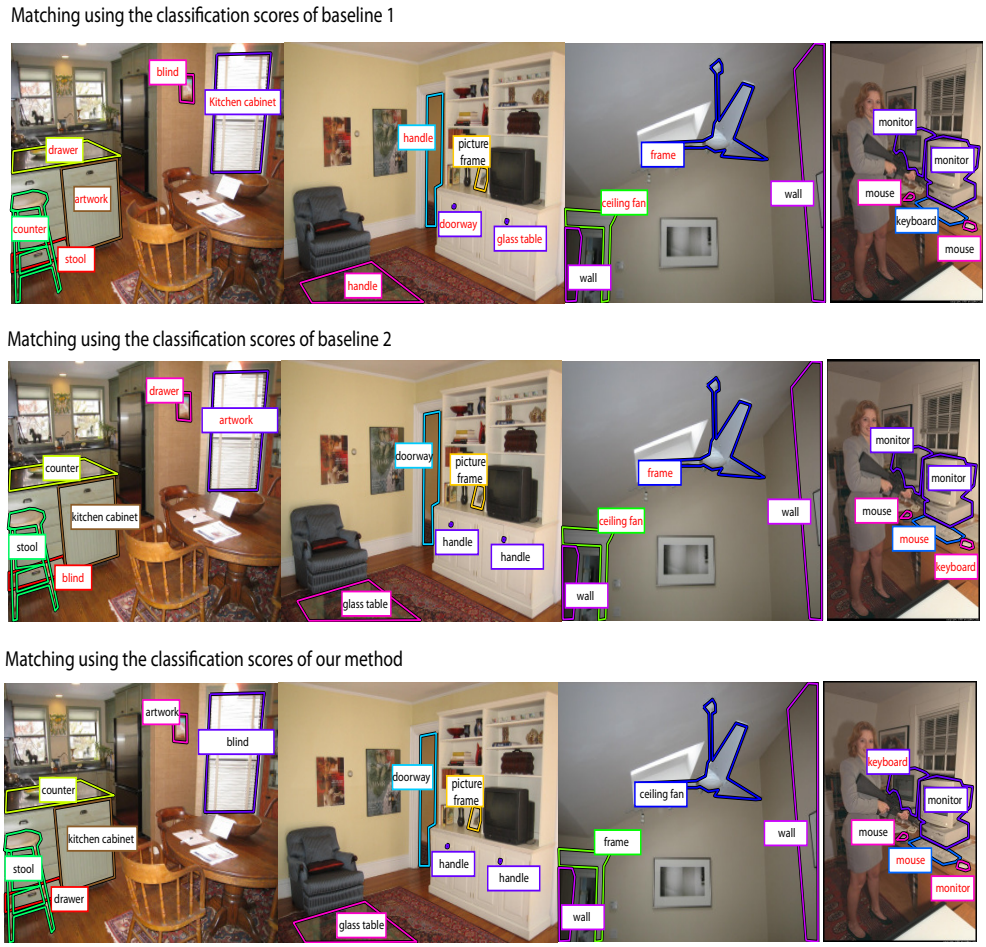

Figure 5.7: Examples showing found correspondence between regions and weak class labels. On each image, regions are depicted by polygons in different colors; the found corresponding class names are surrounded by squares in the same colors. Incorrect correspondence is indicated with red object names. Each column shows comparison on the same image. For the first three images, using classification scores by our method finds better correspondence. This is mainly because many categories such as "artwork" and "ceiling fan" have few or no training examples, so the baseline classifiers cannot learn good models for them. Our method does not work well on the fourth image, because "mouse" and "keyboard" have strong similarity correlation (their similar categories are both labeled as "book" and "box"). One mouse (keyboard) model trained with similarity may be more likely to confuse keyboard (mouse).

CHAPTER 6

CONCLUSION

This dissertation has introduced my work with colleagues on visual recognition, in terms of datasets, features, learning, and visual models.

Our work on dataset collection presents a novel idea to exploit online knowledge resources for object image retrieval, using human compiled data to build object models. We perform experiments on two datasets. Experimental results show the effectiveness of this approach.

We show that text produced by matching an image to a large auxiliary collection of images which have noisy annotations is a surprisingly powerful feature. This feature is somewhat independent of direct visual features. It can be used to enhance any visual feature capable of producing matches, and doing so has consistently improved recognition performance in our experiments with large standard datasets.

We have proposed SIKMA, an algorithm to quickly train an SVM with the histogram intersection kernel using tens of thousands of training examples. We use SIKMA to train classifiers that predict Flickr group membership. This serves as a basis for image similarity: two images that are likely to belong to the same Flickr groups are considered similar.

We develop an opportunistic model of comparative object similarity, which acts as a category dependent regularizer and produces significant improvements in AUC and correspondence for hundreds of categories with few or no training examples. The model is wholly general and should apply to a wide variety of problems.

REFERENCES

[1] J. Mundy, "Object recognition in the geometric era: A retrospective," in *Toward Category-Level Object Recognition*, J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, Eds. Berlin, Germany: Springer, 2006, pp. 3–28.

[2] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2006.

[4] A. Pinz, "Object categorization," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 4, pp. 255–353, 2005.

[5] L. Fei-Fei, R. Fergus, and A. Torralba, "Object recognition short course," 2005-2009. [Online]. Available: http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html

[6] J. Ponce et al., "Dataset issues in object recognition," in *Toward Category-Level Object Recognition*, J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, Eds. Berlin, Germany: Springer, 2006, pp. 29–48.

[7] T. Berg, A. Sorokin, G. Wang, D. Forsyth, D. Hoiem, I. Endres, and A. Farhadi, "It's all about the data," *Proceedings of the IEEE*, vol. 98, pp. 29–48, 2010.

[8] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2003, pp. 409–415.

[9] S. Nene, S. Nayar, and H. Murase, "Columbia object image library (coil-20)," 1996. [Online]. Available: http://www. cs. columbia. edu/CAVE/coil-20. html

[10] C. V. G. at Caltech, "Caltech 5 dataset," 2001. [Online]. Available: http://vision.caltech.edu/archive.html

[11] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," in *Workshop and Special Issue on Generative-Model Based Vision*, 2004, p. 178.

[12] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Pasadena, California, Tech. Rep. 7694, 2007. [Online]. Available: http://authors.library.caltech.edu/7694

[13] N. Dalai, B. Triggs, I. Rhone-Alps, and F. Montbonnot, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[14] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169–2178.

[15] PASCAL, "Pascal voc datasets," 2005-2010. [Online]. Available: http://pascallin.ecs.soton.ac.uk/challenges/VOC/

[16] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[17] W. Gang, Z. Ye, and L. Fei-Fei, "Using dependent regions for object categorization in a generative framework," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1597–1604.

[18] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1794–1801.

[19] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *International Conference on Computer Vision*, vol. 2, 2005, pp. 1458–1465.

[20] A. Berg, T. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 26–33.

[21] J. Mutch and D. Lowe, "Multiclass object recognition using sparse, localized features," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 11–18.

[22] L. Li, G. Wang, and L. Fei-Fei, "OPTIMOL: Automatic Online Picture collecTion via Incremental MOdel Learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[23] T. Berg and D. Forsyth, "Animals on the web," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1463–1470.

[24] F. Schroff, A. Criminisi, and A. Zisserman, "Harvesting image databases from the web," in *International Conference on Computer Vision*, 2007, pp. 1–8.

[25] G. Wang and D. Forsyth, "Object image retrieval by exploiting online knowledge resources," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[26] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[27] L. Von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2004, pp. 319–326.

[28] L. Von Ahn, R. Liu, and M. Blum, "Peekaboom: A game for locating objects in images," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2006, pp. 55–64.

[29] A. Sorokin and D. Forsyth, "Utility data annotation with amazon mechanical turk," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2008, pp. 1–8.

[30] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[31] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of Applied Statistics*, vol. 21, no. 1, pp. 225–270, 1994.

[32] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 465–470.

[33] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, 1998.

[34] T. Kadir and M. Brady, "Scale, saliency and image description," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[35] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector," in *European Conference on Computer Vision*, 2004, pp. 228–241.

[36] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[37] N. Snavely, S. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," in *ACM SIGGRAPH*, 2006, pp. 835–846.

[38] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 524–531.

[39] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," 2007. [Online]. Available: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/

[40] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 506–513.

[41] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[42] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*, 2006, pp. 404–417.

[43] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[44] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1265–1278, 2005.

[45] W. Freeman and E. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, 1992.

[46] J. Koenderink and A. Van Doorn, "Representation of local geometry in the visual system," *Biological Cybernetics*, vol. 55, no. 6, pp. 367–375, 1987.

[47] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or how do I organize my holiday snaps?" in *European Conference on Computer Vision*, 2002, pp. 414–431.

[48] L. Van Gool, T. Moons, and D. Ungureanu, "Affine/photometric invariants for planar intensity patterns," in *European Conference on Computer Vision*, 1996, pp. 642–651.

[49] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3485–3492.

[50] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *ECCV Workshop on Statistical Learning in Computer Vision*, 2004, pp. 1–22.

[51] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[52] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for distributions with applications to image databases," in *International Conference on Computer Vision*, 2002, pp. 59–66.

[53] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *International Conference on Computer Vision*, 2003, pp. 1470–1477.

[54] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, no. 1, pp. 177–196, 2001.

[55] T. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences*, vol. 101, no. 1, pp. 5228–5235, 2004.

[56] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[57] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman, "Discovering objects and their location in images," in *International Conference on Computer Vision*, vol. 1, 2005, pp. 370–377.

[58] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, pp. 145–175, 2001.

[59] I. Biederman, "Recognition-by-components: A theory of human image interpretation," *Psychological Review*, vol. 94, no. 115-148, pp. 32–33, 1987.

[60] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computer*, vol. c-22, no. 1, pp. 67–92, 1973.

[61] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision.*, vol. 1, pp. 55–79, 2005.

[62] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2003, pp. 264–271.

[63] M. Burl and P. Perona, "Recognition of planar object classes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 223–230.

[64] M. Burl, M. Weber, and P. Perona, "A probabilistic approach to object recognition using local photometry and global geometry," in *European Conference on Computer Vision*, 1996, pp. 628–641.

[65] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[66] R. Fergus, P. Perona, and A. Zisserman, "A sparse object category model for efficient learning and exhaustive recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 380–387.

[67] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[68] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, "Spatial priors for part-based recognition using statistical models," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 10–17.

[69] G. Bouchard and B. Triggs, "Hierarchical part-based visual object categorization," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 710–715.

[70] P. Felzenszwalb and D. McAllester, "Object detection grammars," University of Chicago, Chicago, IL, Tech. Rep., 2010.

[71] S. Zhu and D. Mumford, "A stochastic grammar of images," *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 4, pp. 259–362, 2006.

[72] C. Gu, J. Lim, P. Arbeláez, and J. Malik, "Recognition using regions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1030–1037.

[73] B. Russell, W. Freeman, A. Efros, J. Sivic, and A. Zisserman, "Using multiple segmentations to discover objects and their extent in image collections," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1605–1614.

[74] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–23, 2009.

[75] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[76] K. Barnard, P. Duygulu, and D. Forsyth, "Clustering art," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 434–441.

[77] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. Blei, and M. Jordan, "Matching words and pictures," *Journal of Machine Learning Research*, vol. 3, pp. 1107–1135, 2003.

[78] Y. Chen and J. Z. Wang, "Image categorization by learning and reasoning with regions," *Journal of Machine Learning Research*, vol. 5, pp. 913–939, 2004.

[79] Y. Wang, Z. Liu, and J.-C. Huang, "Multimedia content analysis-using both audio and visual clues," *IEEE Signal Processing Magazine*, vol. 17, no. 6, pp. 12–36, 2000.

[80] H. Wactlar, T. Kanade, M. Smith, and S. Stevens, "Intelligent access to digital video: The informedia project," *Computer*, vol. 29, no. 5, pp. 46–52, 1996.

[81] R. Datta, J. Li, and J. Z. Wang, "Content-based image retrieval: Approaches and trends of the new age," in *ACM SIGMM International Workshop on Multimedia Information Retrieval*, 2005, pp. 253–262.

[82] S. Belongie, C. Carson, H. Greenspan, and J. Malik, "Color and texture-based image segmentation using EM and its applications to content based image retrieval," in *IEEE International Conference on Computer Vision*, 1998, pp. 675–682.

[83] D. Joshi, J. Z. Wang, and J. Li, "The story picturing engine: Finding elite images to illustrate a story using mutual reinforcement," in *ACM SIGMM International Workshop on Multimedia Information Retrieval*, 2004, pp. 119–126.

[84] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1075–1088, 2003.

[85] O. Maron and A. Ratan, "Multiple-instance learning for natural scene classification," in *International Conference on Machine Learning*, 1998, pp. 341–349.

[86] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth, "Object recognition as machine translation," in *European Conference on Computer Vision*, 2002, pp. IV: 97–112.

[87] P. Brown, S. D. Pietra, V. D. Pietra, and R. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational Linguistics*, vol. 32, no. 2, pp. 263–311, 1993.

[88] D. M. Blei and M. I. Jordan, "Modeling annotated data," in *Annual ACM SIGIR Conference*, 2003, pp. 127–134.

[89] J. Jeon, V. Lavrenko, and R. Manmatha, "Automatic image annotation and retrieval using crossmedia relevance models," in *Annual ACM SIGIR Conference*, 2003, pp. 119–126.

[90] V. Lavrenko, R. Manmatha, and J. Jeon, "A model for learning the semantics of pictures," in *Annual Conference on Neural Information Processing Systems*, 2003, pp. 1–8.

[91] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos, "Supervised learning of semantic classes for image annotation and retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 394–410, 2007.

[92] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to information retrieval," *ACM Transactions on Information Systems*, vol. 22, no. 2, pp. 179–214, 2004.

[93] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[94] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *International Conference on Machine Learning*, 2007, pp. 759–766.

[95] A. Quattoni, M. Collins, and T. Darrell, "Learning visual representations using images with captions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[96] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning object categories from Google's image search," in *IEEE International Conference on Computer Vision*, 2005, pp. 1816–1823.

[97] D. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision*, 1999, pp. 1150–1157.

[98] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[99] S. Maji, A. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[100] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: A power tool for interactive content-based image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 644–655, 1998.

[101] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, June 2001.

[102] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–535, 1997.

[103] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *International Journal of Computer Vision*, vol. 62, no. 1, pp. 61–81, 2005.

[104] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, p. 433, 1999.

[105] D. Xu, T. Cham, S. Yan, and S. Chang, "Near duplicate image identification with patially aligned pyramid matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–7.

[106] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2559–2566.

[107] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for distributions with applications to image databases," in *International Conference on Computer Vision*, 1998, pp. 59–66.

[108] N. Rasiwasia, P. Moreno, and N. Vasconcelos, "Bridging the gap: Query by semantic example," *IEEE Transactions on Multimedia*, vol. 9, no. 5, pp. 923–938, 2007.

[109] Y. Rui, T. Huang, and S. Chang, "Image retrieval: Current techniques, promising directions and open issues," *Journal of Visual Communication and Image Representation*, vol. 10, no. 4, pp. 39–62, 1999.

[110] R. Datta, D. Joshi, J. Li, and J. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys*, vol. 40, no. 2, pp. 1–60, 2008.

[111] T. Malisiewicz and A. Efros, "Recognition by association via learning per-exemplar distances," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[112] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," in *ACM Multimedia*, 2004, pp. 869–876.

[113] Y. Liu, D. Zhang, G. Lu, and W. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.

[114] M. Swain and D. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[115] C. Jacobs, A. Finkelstein, and D. Salesin, "Fast multiresolution image querying," in *ACM Conference on Computer Graphics and Interactive Techniques*, 1995, pp. 277–286.

[116] K. Barnard and D. Forsyth, "Learning the semantics of words and pictures," in *International Conference on Computer Vision*, 2001, pp. 408–415.

[117] I. Cox, M. Miller, S. Omohundro, and P. Yianilos, "Pichunter: Bayesian relevance feedback for image retrieval," in *International Conference on Pattern Recognition*, vol. 13, 1996, pp. 361–369.

[118] P. Enser, "Pictorial information retrieval," *Journal of Documentation*, vol. 51, no. 2, pp. 126–170, 1995.

[119] A. Frome, Y. Singer, F. Sha, and J. Malik, "Learning globally consistent local distance functions for shape-based image retrieval and classification," in *International Conference on Computer Vision*, 2007, pp. 1–8.

[120] G. Wang, A. Gallagher, J. Luo, and D. Forsyth, "Seeing people in social context: Recognizing people and social relationships," in *European Conference on Computer Vision*, 2010, pp. 169–182.

[121] L. Zhang, Y. Hu, M. Li, W. Ma, and H. Zhang, "Efficient propagation for face annotation in family albums," in *ACM International Conference on Multimedia*, 2004, pp. 716–723.

[122] J. Cui, F. Wen, R. Xiao, Y. Tian, and X. Tang, "EasyAlbum: An interactive photo annotation system based on face clustering and re-ranking," in *SIGCHI Conference on Human Factors in Computing Systems*, 2007, pp. 367–376.

[123] Y. Tian, W. Liu, R. Xiao, F. Wen, and X. Tang, "A face annotation framework with partial clustering and interactive labeling," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[124] L. Bottou, *Large-Scale Kernel Machines*. Cambridge, MA: MIT Press, 2007.

[125] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[126] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated subgradient solver for SVM," in *International Conference on Machine Learning*, 2007, pp. 807–814.

[127] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning*. Berlin, Germany: Springer, 2004, pp. 146–168.

[128] G. Wang, D. Forsyth, and D. Hoiem, "Comparative object similarity for improved recognition with few or no examples," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3525–3532.

[129] S. Maji and A. Berg, "Max-margin additive classifiers for detection," in *International Conference on Computer Vision*, 2009, pp. 40–47.

[130] I. Jolliffe, *Principal Component Analysis*. New York, NY: Springer Verlag, 2002.

[131] J. Kruskal and M. Wish, *Multidimensional Scaling*. Thousand Oaks, CA: Sage Publications, 1978.

[132] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, p. 2323, 2000.

[133] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[134] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Advances in Neural Information Processing Systems*, 2003, pp. 521–528.

[135] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[136] A. Torralba, "Contextual priming for object detection," *International Journal of Computer Vision*, vol. 53, no. 2, pp. 169–191, 2003.

[137] V. P. Ameesh Makadia and S. Kumar, "A new baseline for image annotation," in *European Conference on Computer Vision*, 2008, pp. 316–329.

[138] G. Wang, D. Hoiem, and D. Forsyth, "Building text features for object image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1367–1374.

[139] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," 2001. [Online]. Available: http://www. csie. ntu. edu. tw/cjlin/libsvm

[140] N. Loeff and A. Farhadi, "Scene discovery by matrix factorization," in *European Conference on Computer Vision*, 2008, pp. 451–464.

[141] D. Jacobs, D. Weinshall, and Y. Gdalyahu, "Classification with nonmetric distances: Image retrieval and class representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 583–600, 2002.

[142] R. Fergus, Y. Weiss, and A. Torralba, "Semi-supervised learning in gigantic image collections," in *Advances in Neural Information Processing Systems*, 2009, pp. 1–8.

[143] A. Tversky, "Features of similarity," *Psychological Review*, vol. 84, no. 2, pp. 327–352, 1977.

[144] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1778–1785.

[145] C. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 951–958.

[146] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar, "Attribute and simile classifiers for face verification," in *International Conference on Computer Vision*, 2009, pp. 365–372.

[147] M. Palatucci, D. Pomerleau, G. Hinton, and T. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in Neural Information Processing Systems*, 2009, pp. 1–8.

[148] G. Wang and D. Forsyth, "Joint learning of visual attributes, object classes and visual saliency," in *International Conference on Computer Vision*, 2009, pp. 537 – 544.

[149] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," in *Advances in Neural Information Processing Systems*, 1999, pp. 115–132.

[150] G. Wang, D. Hoiem, and D. Forsyth, "Learning image similarity from flickr groups using stochastic intersection kernel machines," in *IEEE 12th International Conference on Computer Vision*, 2010, pp. 428–435.

[151] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *ACM International Conference on Image and Video retrieval*, 2007, pp. 401–408.

[152] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *IEEE Conference on Computer Vision and Pattern Recognition, Anchorage*, 2008, pp. 1–8.

[153] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. New York, NY: Dover Publications, 1998.